



Demagog

Speaking Text Editor - the best solution



[7.30.422](#)

[7.30.422-x64](#)

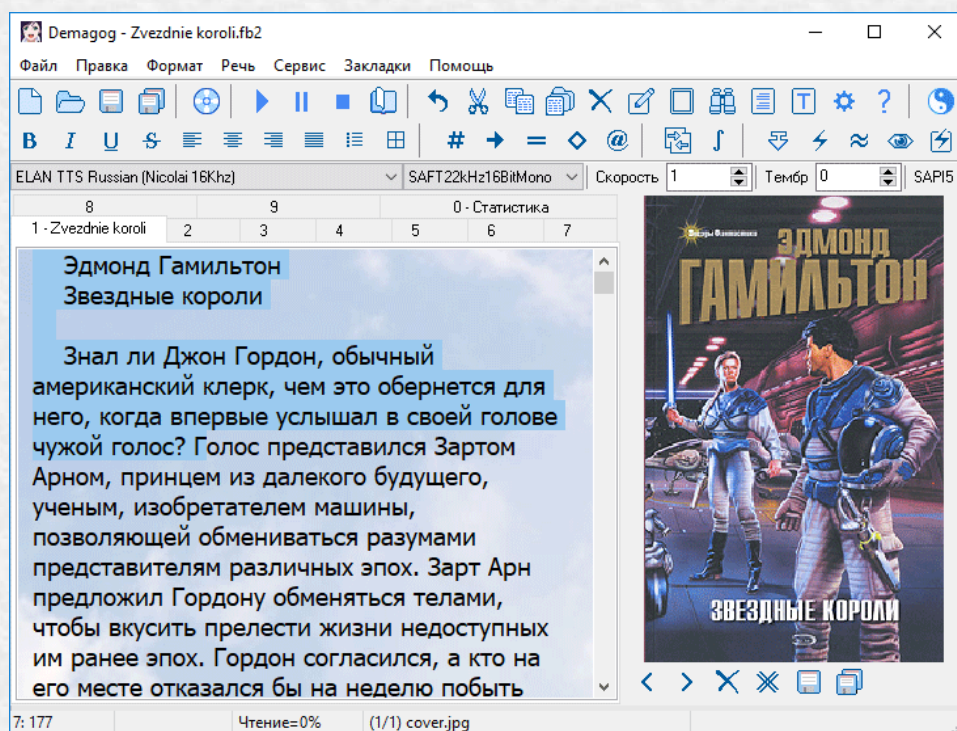
Полная поддержка юникода,
многоязычный интерфейс.
Full Unicode support, multi-
language interface.

[История версий](#)

В дистрибутив включены исходные коды библиотек: для поддержки регулярных выражений; и для встроенного интерпретатора. Библиотеки лицензированы под MPL 2.0.
The distribution includes source code: [SkRegExp](#) to support regular expressions; and [VerySimpleLua](#) to support built-in interpreter. These libraries are licensed under the [MPL 2.0](#).

Говорящий текстовый редактор

© Benedict Lee lotonges@gmail.com n1d3@yahoo.com



[Введение](#)

[1. SAPI4 и Николай Еланович Дигало](#)

[2. SAPI5, три грации и старый друг](#)

[3. Нормальные герои... или повесть об алгоритме словарных замен](#)

[4. Нормальные герои, продолжение... или нет предела совершенству](#)

[5. Свежий взгляд](#)

[6. "Глокая куздра", "четыре четыре" и питон](#)

[7. "Говорит и показывает..."](#)

[8. Чтение многоязычных текстов. Мультилингва](#)

[9. REX-словари. Склонятор. Поиск в тексте](#)

[10. Встроенный интерпретатор](#)

[11. Вызов программы из командной строки](#)

- [12. Подсветка ключевых слов в компоненте RichEdit](#)
[13. Как добавить в программу новый язык интерфейса?](#)
[14. Путь самурая. Demagog в мире Юникода](#)
[15. "Брюки превращаются..." или что такое "фонетический алфавит"?](#)
[16. Формат DXT - "документ Demagog"](#)
[17. Извлечь шляпу из кролика... или текст любой ценой](#)
[18. Великан на дороге, или нейросети для синтеза речи](#)
[19. Сам себе Гутенберг или электронная книга своими руками](#)
[20. Новые горизонты. 64-разрядная версия Demagog](#)
[Заключение](#)

Введение

*Но этот меч, который я даю вам в руки,
 есть лучший из всех мечей, что я раньше сделал.
 Для этой цели удача посетила меня.
 Я сделал это из философских соображений...*

Оружейник Хаттори Хансо

Demagog - это текстовый редактор, который может прочесть вслух загруженный в него текст или записать его в виде аудио-файла. Вот, кратко, его возможности. Полная поддержка Юникода. Чтение текста вслух. Поддержка чтения многоязычных текстов. Конвертация текста в аудио-файл. Пакетная запись аудио. Запись аудио в виде сериала, т.е. с делением на фрагменты заданного размера. Поддержка словарей корректуры произношения популярного формата DIC. Поддержка аналогичных словарей формата REX, основанных на регулярных выражениях. Импорт рисунков из документов MS Word и E-Book. Развитая система поиска и замены в тексте. Подсветка орфографических ошибок, омографов, и близко стоящих похожих слов (опция "Свежий взгляд"). Пользовательские настройки окна редактирования: шрифт, фон или фоновая картинка. Встроенный интерпретатор для создания пользовательских скриптов: например, экспресс-анализа текстов или математических расчетов.

Название программы соответствует ее назначению, и происходит от первоначального значения греческого слова δημαγωγός - "говорящий с народом". Программа не требует установки и поставляется уже готовой к использованию. Для этого надо запустить исполняемый файл Demagog.exe. Для удаления программы достаточно удалить с компьютера папку Demagog со всем ее содержимым. Программа не изменяет никакие системные файлы, не содержит рекламы, не собирает и не передает личные данные пользователей. Пользование программой - бесплатное. В программу встроена Справка на русском, украинском, английском, эсперанто.

ЭТА ПРОГРАММА РАСПРОСТРАНЯЕТСЯ "КАК ЕСТЬ". НИКАКИХ ГАРАНТИЙ, ЯВНО ВЫРАЖЕННЫХ ИЛИ ПОДРАЗУМЕВАЕМЫХ. ВЫ ИСПОЛЬЗУЕТЕ ЕЕ НА СВОЙ СТРАХ И РИСК. АВТОР НЕ НЕСЕТ ОТВЕТСТВЕННОСТИ ЗА ПОТЕРЮ ДАННЫХ, ПОВРЕЖДЕНИЕ, ПОТЕРЮ ПРИБЫЛИ ИЛИ ЛЮБЫЕ УБЫТКИ ПРИ ИСПОЛЬЗОВАНИИ ИЛИ НЕПРАВИЛЬНОМ ИСПОЛЬЗОВАНИИ ЭТОГО ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ.

THIS PROGRAM DISTRIBUTED "AS IS". NO WARRANTY OF ANY KIND IS EXPRESSED OR IMPLIED. YOU USE IT AT YOUR OWN RISK. THE AUTHOR WILL NOT BE LIABLE FOR DATA LOSS, DAMAGED, LOSS OF PROFITS OR ANY KIND OF LOSS WHILE USING OR MISUSING THIS SOFTWARE.

Первой опубликованной в Инете была версия 1.08.020 beta, еще не имевшая всех нынешних функциональных возможностей. 28 октября 2007 года - день рождения программы Demagog.

Фрагменты исходного текста Demagog были использованы разработчиками погодного робота [Meteonova](#), созданного под руководством кандидата географических наук Александра Королькова.

В 2009 г. Demagog занял 3-е место в конкурсе Soft.Mail.ru "Самые популярные программы года" в номинации "Текст".

Одно из авторитетнейших компьютерных изданий - выходящий на 15 языках журнал "CHIP" (№5/2014), назвал Demagog в числе наиболее известных [программ для управления речевыми движениями](#).

Издание на Google books: "[250 лучших бесплатных программ](#)" так же включило Demagog в свой список, раздел "Преобразование текста в звук".

Ссылку на Demagog можно обнаружить на множестве интернет-ресурсов, однако новые версии надо скачивать только с этой моей странички.

| | |
|------------------------------------|---|
| Demagog.zip | Дистрибутив программы. Для установки достаточно распаковать архив в какое-либо место на вашем компьютере. |
| Demagog-x64.zip | Дистрибутив 64-разрядной версии программы. См. главу " Новые горизонты... " Для установки достаточно распаковать архив в какое-либо место на вашем компьютере. Добавлена интеграция с языком Python. По умолчанию используется встроенный интерпретатор. Но, если на вашем компьютере уже имеется установленный Python, то можно подключиться к нему. Такая возможность полезна, когда вы работаете с Python, в который добавлены различные внешние библиотеки. |
| Demagog-x64-Silero | Экспериментальная сборка для озвучивания текстов голосами от компании Silero . !! Этот дистрибутив, при распаковке, требует наличия 1.4 Гб свободного места на жестком диске. Кроме собственно Demagog, в нем содержатся нейросетевые модели для 6 языков и модуль pytorch для работы с ними. Инструкция в pdf-формате находится внутри архива. См. также главу " Великан на дороге... ", параграф "Demagog-x64-Silero". |
| numbers.zip | Словари для русского и английского языка, преобразующие числа в их текстовое представление. |
| Text Mining Tool | Портативная утилита конвертации PDF Word -> TXT |

| | |
|--|--|
| Text Mining Tool | Портативная утилита конвертации TSV, Word > TXT. |
| RU-YO-orfo.zip | Значительно более полный вариант русской орфотаблицы. В отличие от поставляемой с дистрибутивом, эта орфотаблица содержит и варианты с буквой "ё". |
| NICOLAI | Русский голосовой движок Elan Nicolai, в просторечии - "Николай". Давно устарел и больше не поддерживается разработчиком. Но всё еще популярен в кругу фанатов. Выгодно отличается от более современных голосов размером дистрибутива, всего 11 Mb. Работает под SAPI4 и SAPI5. |
| SAPI4 | В ОС Windows, начиная с версии XP и выше - по умолчанию установлена голосовая система SAPI5. Если хотите, здесь вы можете дополнительно скачать SAPI4. Голосовые системы SAPI4 и SAPI5 не замечают друг друга и могут работать на одном компьютере. |
| Словарь "Микеланджело" | Словарь для движка "Николай". Свыше 37700 правил. Никаких гарантий, что он не содержит ошибочных или излишних правил, не дается. !! Замечание о словарях для движка Николай. Словари *.dic для Николая, созданные некогда для SAPI4, работают в Demagog и под SAPI5, при условии, что в "Сервис - Общие настройки..." - Чтение" стоит галочка "Игнорировать теги SAPI5". Тогда символы < > в словаре воспринимаются правильно: как знаки ударения. |
| Lat2Cyr.zip | Словарь "транслитер" для преобразования латиницы в кириллицу. |
| opusenc.zip | Неплохой аудио-конвертер, по мнению его разработчиков - лучший из всех. Он не входит в дистрибутив Demagog, но может быть помещен в его рабочий каталог и подключен через меню "Сервис - Общие настройки..." - Аудио - Custom Encoder". Командную строку пользователь указывает вручную, она будет запомнена в настройках. Что-то вроде: opusenc.exe --bitrate 48 %1 %2 . Символы %1 и %2 - это условные имена входного и выходного аудиофайлов; реальные имена сгенерирует и подставит в командную строку сам Demagog. Тип выходного файла надо бы указывать "родной": opus, но подойдет и ogg. Тогда полученное аудио будет распознаваться и проигрываться Winows Media Player'ом. |
| fastencc.zip | Аудио-конвертер в mp3, более быстрый, чем lame. Руководство пользователя - в архиве. Пример вызова как "Custom Encoder": fastencc.exe %1 %2 -br 48000 |
| flac.zip | Аудио-конвертер в звуковой формат flac . Версии для win32-64. Руководство пользователя - в архиве. Пример вызова как "Custom Encoder": flac.exe %1 -o %2 |
| sox.zip | SoX - Sound eXchange . Свободно распространяемый консольный аудио-редактор с великим множеством функций. Эта сборка не требует инсталляции, просто распакуйте zip-архив в любое место на вашем компьютере. Руководство пользователя (очень большое!) - в архиве. Пример вызова, как "Custom Encoder": полный путь\sox.exe -G --multi-threaded -q %1 -C 64.01 %2 tempo -s 1.1 |
| ffmpeg.zip | Утилита командной строки для конвертирования видеофайла из одного формата в другой. С её помощью можно также захватывать видео в реальном времени с TV-карты. Для консольного приложения имеет необычно большой размер. Руководство пользователя (весьма объемистое, на английском) в архиве. |
| mediainfo.zip | Кросс-платформенная программа с открытым исходным кодом для получения технической информации об аудио и видеофайлах различных форматов (в общей сложности 46 медиаформатов), а также о форматах изображений. Руководство пользователя (на английском) в архиве. |
| converter_lua.zip от пользователя balaamster | Скрипт для Demagog, предназначенный для выполнения утилит ffmpeg и mediainfo под управлением Demagog. Архив необходимо распаковать в папку с Demagog. В папку fon скопировать mp3-файлы с фоновым звуком. ffmpeg.exe должен лежать также в папке с Demagog. Фоны для каждой серии выбираются автоматически, случайным образом. Порядок действий: сначала создаём серию в mp3, wav или ogg; запускаем "converter.lua"; выбираем папку, где лежит серия и формат, в котором создана серия; выбираем целевую папку - целевой формат - битрейт с фоном или без; выбираем желаемый темп речи. Смотрим выбранные параметры, принимаем решение. |
| converter_bat.zip от пользователя balaamster | bat-файл, предназначенный для выполнения утилит ffmpeg и mediainfo под управлением Demagog. - Распаковываем conv.bat в папку Demagog. (mediainfo.exe и ffmpeg.exe должны также лежать там, а также папка fon с фоновыми mp3-файлами - например: fon.mp3); - В настройках аудио-конвертера: прописываем %~conv.bat в качестве исполняемого файла. Порядок параметров такой: путь_входного_файла путь_выходного_файла битрейт темп реверберация (true или false) громкость_фона (от 0.1 до 1.0) Например: %1 %2 64 1.0 true 0.1 64 - битрейт, 1.0 - темп, true - с реверберацией, 0.1 - громкость фона; формат прописываем mp3 или ogg . Назначение - то же самое, что у скрипта, упомянутого выше. Но, не требует от пользователя никаких умственных усилий по выбору параметров. Загрузил в Demagog текст, нажал кнопку "Записать аудио" или клавишу Ctrl+M. И... пошло-поехало само собой. Пример звучания , полученный следуя этим инструкциям |

| | |
|---|---|
| <p>"Демагог одной кнопкой" от пользователя tonio_k</p> <p>Зеркало 1 Зеркало 2 Зеркало 3</p> | <p>пример озвучивания, полученный, следуя этим инструкциям.</p> <p>Инструкция:</p> <ol style="list-style-type: none"> 1. Открываем файл с книгой 2. Нажмаем Ctrl+F2 3. В открывшемся окне запускаем: 00_ПЕРВИЧНАЯ ОБРАБОТКА.lua 4. Ждем информационного окна о завершении подготовки книги 5. Можно приступить к чтению вслух или записи в mp3 <p>Примечание. На компьютере должен быть установлен синтезатор голоса IVONA Maxim. Словари ориентированы исключительно на него!</p> |
| <p>DemagogAndYandexTTS.zip</p> <p>Набор скриптов для озвучивания книг через сервис Yandex TTS. Разработчик: balaamster Идея и тестирование: tonio_k</p> <p>---</p> <p>Замечание. Эти скрипты предоставляют доступ к старому, бесплатному демо-сервису Яндекса. Он давно не обновляется, но все еще доступен в Инете. Самые качественные голоса: Флипп и Алена исключены из него и переведены в новый, платный сервис Yandex SpeechKit</p> | <p>Инструкция от balaamster</p> <ol style="list-style-type: none"> 1. Для настройки голоса запускаем через Ctrl+F2 YaTTS_configure.lua выбираем голос, темп речи, эмоциональную окраску, количество потоков для скачивания, длительность аудиофрагмента. 2. Для получения mp3-файлов: <ul style="list-style-type: none"> - открываем книгу во вкладке - запускаем через Ctrl+F2 YaTTS.lua - указываем папку для сохранения mp3-файлов озвученной книги - ожидаем, пока закроется чёрное окно консоли (завершится процесс скачивания, прогрессбар сверху окна) <p>В ранее выбранном каталоге находим озвученную книгу. Если процесс скачивания необходимо прервать, в окне консоли нажимаем Ctrl+C. (временные txt- mp3-файлы сохраняются в каталоге %temp%\имя_файла_книги\, в каталоге для готовых mp3-файлов при этом ничего не появится. Если в процессе скачивания возникнут ошибки - сетевой сбой, ошибка работы скрипта, то появится сообщение о несоответствии количества текстовых файлов сериала и полученных mp3-файлов. В Проводнике откроется каталог, в котором находятся временные txt-файлы сериала, полученные mp3-файлы (подкаталог audio) и лог-файл (подкаталог log).</p> <p>Процесс скачивания можно продолжить, заново запустив скрипт на вкладке с книгой и ответив на запрос: ДА - продолжить преобразование с последнего необработанного файла.</p> 3. Для озвучивания слова/фразы/текстового фрагмента(не более 1000 символов): <ul style="list-style-type: none"> - во вкладке с книгой выделяем фрагмент текста, который необходимо озвучить - запускаем через Ctrl+F2 YaTTS_speak.lua - ожидаем, пока закроется чёрное окно консоли (завершится процесс скачивания) <p>Откроется консоль Power Shell, в которой воспроизведётся выделенный фрагмент. Если выделенный фрагмент не изменён и его надо прослушать повторно, то можно запустить скрипт ещё раз, при этом повторное скачивание производиться не будет. (или можно самостоятельно открыть файл %temp%\speak_tmp\audio\speak.mp3)</p> <p>Для использования скрипта необходимо наличие ffmpeg.exe в папке с Demagog. Скачать можно тут. Содержимое архива Demagog+YandexTTS.zip распаковать в папку с Demagog.</p> <p>В YaTTS.lua переменные destination, clean_start и delete_source сделаны глобальными. Это позволяет запускать YaTTS.lua из других скриптов совсем без диалоговых окон. Например:</p> <pre>ind = WActive() WFilter(ind, ind, HomeFolder('dic')..'70_Яндекс+ОМО.dic') destination = 'D:\audiobook\book1\' clean_start = true --очистить папку с временными файлами, перед началом обработки delete_source = true --удалить папку с врем. файлами после успешного завершения dofile(HomeFolder('_Tests_')..'YaTTS.lua')</pre> <p>Этот скрипт применит словарь "70_Яндекс+ОМО.dic" к активной вкладке и запустит скачивание с Яндекса без диалоговых окон. Также имеются глобальные переменные artist и album. Через них задаются соответствующие id3-тэги. По умолчанию значения artist = "Yandex TTS", album- имя файла открытой книги. (title - неизменяемый, значение - номер аудиофрагмента).</p> <p>Можно задать свои значения:</p> <pre>artist = "Филипп"</pre> |

| | |
|----------------------------|---|
| | <pre>artist = "Ерма" album = "Моя любимая книга" dofile(HomeFolder('_Tests_')..'YaTTS.lua')</pre> |
| Source.zip | Исходный код одной из ранних версий Demagog. |
| Форум | Форум о преобразовании текста в речь. Ветка, посвященная программе Demagog. |

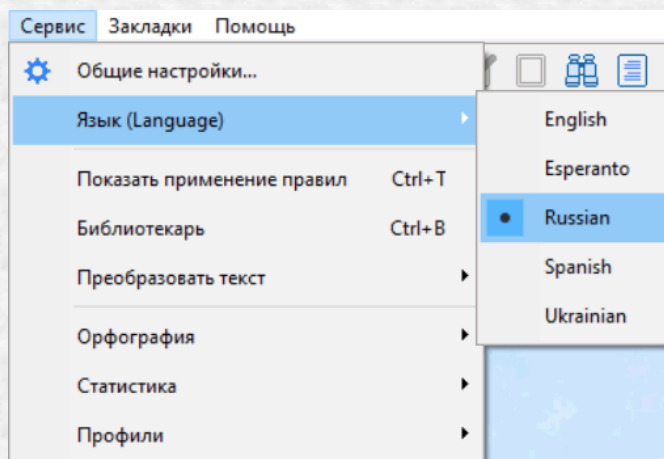
<<

1. SAPI4 и Николай Еланович Дигало

Многие слышали об их существовании, а кое-кто даже ими пользуется. Программы-читалки текстов, берегущие наши глаза. "TextAloud", "Speaking Notepad", "Балаболка", уже полузабытая знаменитая "Говорилка"..., и другие. Считается, что создать нечто подобное - это высший программистский пилотаж. На самом деле - ничего такого сложного 😊 Здесь я опишу упражнение на языке Delphi, дающее вполне удовлетворительный результат. Тестировалось под ОС Windows XP, Vista, 7, 8, 10. Как это смотрится под "десяткой" вы уже видели.

Всплывающие подсказки под каждой кнопкой объясняют их назначение, при этом в правой части строки состояния внизу дается развернутая подсказка. Когда курсор мыши находится над текстом, в строке состояния показывается размер текстового файла. Остальные данные в строке состояния, по порядку: позиция текстового курсора; код символа за курсором (в данном случае - это пробел); название файла показываемой картинки и их количество; коэффициент увеличения размера текста, задаваемый вращением колесика мыши.

У программы многоязычный интерфейс. Его описание содержится в файлах типа *.ln. Добавив новый файл "языкового ресурса", мы, тем самым, добавим новый язык в меню.



Итак, начнем с начала. Задача "синтеза речи", т.е. превращение электронного текста в звук давно решена средствами Windows. В Win 98, NT, 2000 установлен т.н. "менеджер голосовых функций" - Speech API версии 4.0. Сокращенно: SAPI4. В Win XP, Vista, 7, 8, 10 - SAPI5. Эти голосовые системы абсолютно не совместимы, но друг другу не мешают и могут сосуществовать на одном компьютере.

Далее. Чтобы компьютер заговорил, на нем должна быть установлена хоть одна Text To Speech Engine. В переводе на русский: "голосовой движок". Остается соорудить программную оболочку, для распознавания имеющихся на компьютере голосовых движков, загрузки текстового файла и передачи его выбранному движку для чтения вслух!

Для работы в режиме SAPI4 Demagog использует соответствующий низкоуровневый (COM) интерфейс. Очень ясно (с примерами и без лишних деталей) необходимые функции SAPI4 описаны в статьях [Брайана Лонга](#) (для тех, кто читает на английском) и в книге Дениса Буторина "MS Agent и Speech API для Delphi". Первую половину книжки, про MS Agent'a (создание мультяшных "помощников") смело пропускаем. Дальше читаем внимательно. Поэкспериментируйте с демо-проектами с прилагаемого к книге CD. Стараться понять до конца, как это работает, не нужно, главное - усвоить последовательность, в какой должны объявляться TTS-интерфейсы. Для хранения текста используется компонент RichEdit. Кнопки добавляются "по вкусу". Я использовал компоненты ActionList и ImageList для централизованного построения графического интерфейса.

Наиболее популярным русским голосовым движком под SAPI4 был Nicolai, разработанный французской компанией Elan (позже сменившей название на Asarella). Он давно устарел и больше не поддерживается разработчиком, но все еще популярен у фанатов. Его последнюю версию Elan Nicolai 5.1 до сих пор можно найти на просторах Инета. Отмечу, что эта версия работает и под SAPI5.

При написании Demagog я придерживался принципа "минимализма". Классическое "выпадающее" меню и кнопки для типичных действий, объединенные в основную и дополнительную "панели инструментов". Наличествуют стандартные функции текстового редактора, включая развитую (и очень шустрю) систему поиска/замены в тексте; а также приведение текста к аккуратному "литературному" виду.

Помимо этого, Demagog понимает и автоматически преобразует в ANSI (кириллица Windows) старые кодировки: DOS (из уважения к DOS) и KOI8-R (из уважения к Максиму Мошкову).

Demagog поддерживает словари корректировки произношения популярного формата DIC, а также словари формата REX на основе регулярных выражений.

Вместо чтения вслух Demagog может ускоренно записать аудио-файл типа WAV, MP3, OGG, WMA или MP4; звук при этом не слышен. Для конвертации WAV в другие форматы Demagog вызывает соответствующие внешние программы, они входят в установочный комплект, поскольку бесплатные и свободно распространяемые. Пользователь может настроить вызов любой другой внешней программы конвертации, допускающей запуск из

пользователь может настроить вызов любой другой внешней программы конвертации, допускающей запуск из командной строки.

Поддерживается импорт рисунков из открываемых документов MS Word (doc, docx, rtf) и электронных книг: Fiction Book (fb2), Electronic Publication (epub).

Ранние версии с поддержкой одного только SAPI4 обошлись без использования сторонних компонентов и прекрасно скомпилировались на Turbo Delphi 2006 Explorer.

<<

2. SAPI5, три грации и старый друг

С распространением Windows XP, а затем Windows 7 и т.д. разработчики говорящих программ отложили в сторону книгу Дениса Буторина, и взялись за статьи Брайана Лонга, в которых рассказано, [как заставить Delphi понимать SAPI5](#). В Сети можно найти и другие, аналогичные, учебные примеры.

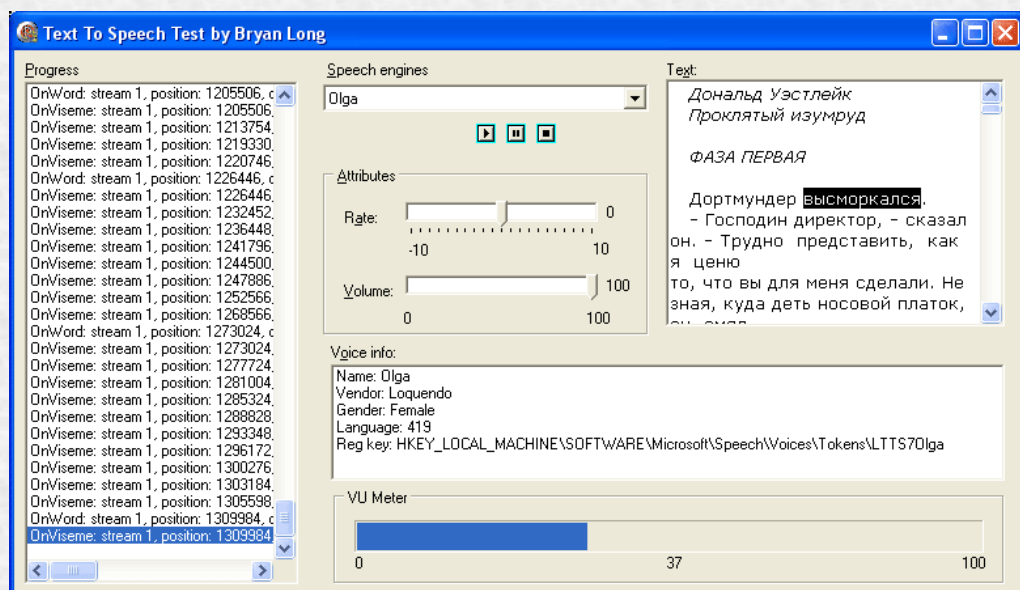
Для поддержки SAPI5 Demagog использует компоненты, импортируемые из Microsoft Speech Object Library, входящей в состав Windows. Для Win 7 и выше, номер версии библиотеки - 5.4.

"Component - Import Component". "Import a Type Library". В открывшемся диалоге находим "Microsoft Speech Object Library". Назначаем имя страницы на палитре компонентов: SAPI. Убеждаемся, что стоит галка на Generate Component Wrappers". Далее будет единственный вариант "Create Unit", его имя будет SpeechLib_TLB.pas. Не забываем, куда он лег! В окне Project Manager на пункте "No project group" правой клавишей мыши открываем меню и выбираем "Add New Project...", далее: "Package". Появится вкладка "Package1.bpl", с присоединенными папками. На папке "Contains" правой клавишей раскрываем меню, "Add...". Указываем на созданный юнит. Снова правой клавишей на "Package1.bpl", "Install".

Появится сообщение с перечнем установленных компонентов. ОК. Если затем мы создадим новую форму, то на палитре компонентов увидим, среди прочих, страницу "SAPI" с новыми компонентами!

Продолжим читать дальше статью Б. Лонга - теперь приведенные в ней примеры будут успешно компилироваться.

Для SAPI5 имеются русскоязычные голосовые движки: Катерина-2 от Next Up (49Mb), Alyona 2.210 от Asapela Group (170Mb), Olga от Loquendo (85Mb). Расскажу подробнее о каждой дамочке. Испытания проводились на [демонстрационной SAPI5-совместимой читалке](#), написанной Брайаном Лонгом.



Катерина-2. Первая версия голоса Katerina до сих пор гуляет по Сети и скачивать ее очень не советую. Она изначально говорит тягуче-медленным голосом законченной наркоманки; дистрибутив содержит в придачу "лекарство", приводящее Катю в чувство. Кроме того, движок плохо исполняет команду SAPI5: SpVoice.Skip('Sentence',MaxInt) - в результате после *останова* чтения длинного (за сотню килобайт) текста, Катюня впадала в ступор минуты на две. Катерина-2 свободна от указанных недостатков, а также делает меньше ошибок в ударениях.

Алена 2.210. Предыдущая версия движка прославилась тем, что не работала под низкоуровневым интерфейсом SAPI4, выдавая фатальную ошибку "Floating point division by zero". Под SAPI5 перед началом чтения длинного текста делает долгую паузу. Кроме того, полностью игнорирует команду SpVoice.Skip().

Ольга. Под SAPI5 выдержала экзамен без замечаний. (Под SAPI4 отсутствует слежение за чтением и регулировка скорости чтения. Нормальный темп чтения будет, если скорость выставить в 0). Интонации у голоса - слегка прибалтийские.

Николай Еланых. Наш старый друг везде прекрасно себя чувствует, хочешь под SAPI5, хочешь под SAPI4. Конкурентов среди русских мужских голосов пока нет. Разные камни в него бросали (хрипат, монотонен, и т.п.), а вот моя знакомая, впервые услышав его чтение, воскликнула: "Какой приятный голос!" Может потому, что *ритмика фразы у него - русская*. (Вряд ли это случайно, ибо в основу движка положены разработки Санкт-Петербургской лаборатории фонетики).

Образцы звучания упомянутых, (а также других, более новых) голосов приведены ниже. Читался отрывок из романа А.Дюма "Граф Монте-Кристо".

| Голос | Аудиофайл | Применялись ли словарные замены |
|---------------------------------------|----------------------------------|--|
| Acapela Elan Nikolai 16 kHz | МК-Nikolai.mp3 | Да (\$Дантес=Дантэ<с *смотрите,=смотри<те, замке=за<мке *гляды*=гля<ды поднял*=подня<л хозяин*=хозя<ин вот=во<т *нибудь=нибу<дь) |
| Next Up Katerina-2 | МК-Katerina2.mp3 | Нет |
| Acapela Alyona 2.210 | МК-Alyona.mp3 | Да (\$Лево=лево смотрите=сматрите губам=губ'ам) |
| Loquendo Olga | МК-Olga.mp3 | Да (\$Дантес=Дант^ес смотрите=смотр^ите тартан*=тарт^ан выстрел*=в^ыстрел) |
| MS Irina (входит в состав Windows 10) | МК-Irina.mp3 | Да (губам=гъубам смотрите=сматрите отдаленного=отдалённого пушечного=пушичного) |
| MS Pavel (входит в состав Windows 10) | МК-Pavel.mp3 | Да (губам=гъубам смотрите=сматрите отдаленного=отдалённого пушечного=пушичного) |
| IVONA Tatyana | МК-Tatyana.mp3 | Да (отдаленного=отдалённого) |
| IVONA Maxim | МК-Maxim.mp3 | Да (смотрите=сматрите отдаленного=отдалённого) |

Теперь начинаем творить собственную читалку под SAPI5. Правда вы (а также я) - не первые. Нас, братцы мои, опередили! Гляньте [сюда](#). Двухязычный синтезатор речи ЕХО Игоря Паламара, на основе статьи Брайана Лонга. Прилагаются исходные тексты программы и сама статья. Кстати, полезное упражнение: удалить из ЕХО все лишнее, оставив работу только с одним языком. Обратите еще внимание, как в ЕХО реализовано блочное чтение теста. Поучительно, не правда ли?

Итак, Demagog отныне поддерживает SAPI5, благо было, на чей опыт опереться. Конечно, я не списывал все буквально у Брайана Лонга. К примеру, он употребил для останова чтения команду SpVoice.Skip('Sentence',MaxInt), что привело к большим неприятностям с некоторыми движениями (см. выше). Вот как правильно: SpVoice.Speak("",SVSPurgeBeforeSpeak).

Замечу, что для компиляции проекта уже не получится использовать Turbo Delphi 2006, где отключена опция импорта компонентов.

<<

3. Нормальные герои... или повесть об алгоритме словарных замен

*Ходы кривые роет подземный умный крот.
Нормальные герои всегда идут в обход!*

Глупцы, героев строя, бросаются вперед.

*Нормальные герои - всегда наоборот.
В обход идти, понятно, не очень-то легко.
Не очень-то приятно и очень далеко.
Зато так поступают одни лишь мудрецы,
Зато так наступают одни лишь храбрецы.
И мы с пути кривого ни разу не свернем!
А, надо будет - снова пойдем кривым путем!*

Песенка из кинофильма "Айболит-66"

"Ну, хорошо", - скажет читатель, забредший на эту страничку, - "Списал у одного - плагиат, списал у нескольких - компиляция, списал у многих - диссертация. А свои-то идеи у автора есть?" На этот вопрос отвечаю положительно.

Что самое главное для голосового движка? Словарь коррективки произношения, ибо учесть все нюансы живого языка никаким гением программирования не под силу. Создатели голосовых движков иногда предусматривают в своих творениях систему поддержки корректировочных словарей, а иногда - нет. Кому как вздумается и кто во что горазд. Видимо поэтому, Антон Рязанов, автор одной из первых работоспособных программ чтения текстов - "Говорилки" придумал универсальный формат словарей произношения, т.н. dic.

```
# Так выделяются комментарии
рыцарский замок=рыцарский за<мок
нет=не<тъ
$Варя=Ва<ря
# #=номер
$$100=сто долларов
# Звездочки справа и/или слева обозначают
# так называемое "сравнение по маске"
туник*=туни<к
*графии=гра<фии
*автобус*=авто<бус
$Серов*=Се>ро<в
*го гнезда=го гнезда<
*, то есть=, то<есть
```

Принцип понятен. При этом, в первую очередь рассматриваются правила с самыми длинными левыми частями.

Фраза "Иван Петрович **Серов** - профессор этно**графии**, нашил в кармане **туники** последние **\$100** и решил ехать на **автобусе**" после подстановок примет вид: "Иван Петрович **Се>ро<в** профессор этно**гра<фии** нашил в кармане **туни<ки** последние **сто долларов** и решил ехать на **авто<бусе**".

и \$ - соответственно, знаки комментария и учета регистра букв при заменах. Чтобы употребить их, просто как символы, их надо удвоить. То же относится и к символу *. Иначе звездочки справа и/или слева обозначают т.н. *сравнение по маске*: оно считается удачным, если обеспечено совпадение с указанной частью слова. Под одну маску могут подходить разные, но частично совпадающие слова. Знаки < > - основное и вспомогательное ударения *для движка Nicolai*. Для других движков знаки ударения - другие или их может не быть вовсе. Тогда эффект ударения можно получить удвоением гласной и т.п. В общем, свобода и воля. Универсализм.

Алгоритм замен очень прост и интуитивно понятен. Каждое правило из словаря применяется ко всему тексту. Правила применяются в порядке убывания длин их левых частей. Т.е. первым будет проверяться правило с самой длинной левой частью. Это логично: сперва в тексте должны заменены самые большие фрагменты. Правила с левыми частями одинаковой длины применяются в том порядке, в каком они расположены в словаре. Как только обнаружено совпадение, так тут же в тексте делается замена и т.д. до конца текста.

В жизни это выглядело бы так. Пусть нам нужно перевести с английского некую интересную книжку. В руках у нас англо-русский словарик, слов эдак на 20 тысяч. Берем 1-е слово из словаря. Читаем книжку от начала до конца в поисках этого слова. Затем переходим ко 2-му слову в словаре и снова читаем книжку целиком. И т.д. пока словарь не закончится.

Слышу голос читателя: "БРЕД!!!"

Согласен. Именно такую бредятину часто пишут программисты, уповая на мощь современных компьютеров. Это называется "прямой перебор".

Те, кто пользовался "Говорилкой", заметили, что она делает легкие паузы через каждые 4-5 предложений. Именно такими мельчайшими порциями программа читает текст. Словарные замены занимают в этом случае доли секунды. Но заминки в чтении уже ощутимы.

Можно заменить функции поиска и замены (простите за каламбур) их ассемблерными аналогами. Есть еще несколько дополнительных технических приемов для снижения трудоемкости прямого перебора. В результате всё будет работать в несколько раз быстрее, нежели в описанной выше бесхитростной реализации. А можно пойти другим путем.

"Швейная машинка Зингера отличается тем, что отверстие для нитки расположено на конце иглы". Здесь мы рассмотрим алгоритм, отличающийся тем, что проверяются все слова из текста в поисках совпадений в заранее составленной хеш-таблице.

Как, на самом деле, мы переводим текст со словарем? Мы смотрим сперва в книгу, а не в словарь! Взявши очередное слово из текста, мы ищем его в словаре. Благодаря алфавитным высекам на его страницах поиск нужного слова занимает секунды.

Еще лучше сказочная ситуация, когда в книге над каждым словом был бы мелкими цифрами напечатан номер страницы, где его искать в словаре. *По сути, именно это я и сделал.*

Вспомним, что каждый символ текста имеет в компьютерном представлении *числовой код*. Комбинируя (специальным образом) коды всех букв в слове, я получаю *адрес* - номер строки *хеш-таблицы*, где его искать. Как составлена хеш-таблица? Обработкой файла словаря: для каждого правила вычисляется адрес его первого слова. Разумеется, может быть несколько правил с одинаковым первым словом (видеть | видеть звезды=видеть звёзды | видеть стены=видеть сте<ны | видеть цвета=видеть цвета<). Тогда в одну строку хеш-таблицы сядет целое словарное гнездо.

Правила, попавшие в одно словарное гнездо, необходимо расположить в порядке убывания длин их левых частей. Тем самым, хоть и не напрямую, но выполняется требование стандарта словарей DIC о порядке применении правил к тексту.

Правила со звездочками обрабатываются наравне с остальными, звездочка - тоже "буква". В результате, каждое правило находится строго на своем месте, откуда будет взято в нужный момент.

Каждое слово может быть найдено в хеш-таблице как одиночное, или как первое в словосочетании. Если, вычислив адрес слова, мы в хеш-таблице обнаружили пустую строку - значит, этого слова в словаре нет. Поехали дальше? Не будем спешить. Не нашли слова "похлебка"? Перебираем все его "огрызки", отбрасывая по одной букве справа и слева и добавляя звездочку с той стороны, где отсечены буквы. Для каждого варианта вычисляем адрес. Ищем. Мимо? Следующий огрызок. Наконец-то! *охлеб* | *охлеб*=охлёб. Делаем замену и проверяем следующее правило в этом же словарном гнезде. Когда проверка гнезда будет закончена, то перейдем к следующему слову в тексте.

Проверка *словосочетания* на совпадение с правилом из словаря проводится сравнением по маске *каждого* слова в левой части правила со словами, входящими в словосочетание. В результате получилось расширенное толкование правил со звездочкой. Она может находиться не только в конце и/или начале левой части правила, а в конце и/или начале *любого слова* в левой части правила. Например, чтобы задать произношение фраз: "У ворот города началась битва", "Изменников повесили на воротах города", "Над воротами города реяли стяги", достаточно одного правила: *ворот* города=ворот<т го<рода*.

Можно даже обрабатывать отдельно стоящие звездочки, как элементы шаблона. Если левая часть правила состоит только из звездочек, (быть может, разделенных пробелами), то все звездочки считаются просто символами. Сравнение с текстом производится не по маске, а на простое равенство строк.

***=...новый раздел...

- заменяется строка ***

*=

- заменяется (на пустую) строка *

В изначальном формате dic-словарей для этих замен приходилось добавлять в маску еще по одной звездочке слева и справа: ** * ***=три звездочки | ***=одна звездочка. Выглядит странно, но и такие варианты легко обрабатываются, отбрасыванием крайних звездочек. Маски *** нет в хеш-таблице? Дополнительно проверим просто * и все дела.

Остается рассмотреть случай, когда *отдельно стоящая звездочка* соседствует с "нормальными" частями шаблона и имеет равные с ними права. Неужели это будет работать?! Ведь под маску из одной только звездочки подойдет

любое слово в тексте! успокойтесь и дышите глубже. По правилу: * **то есть**=то<есть в тексте будет найдено **любое слово**, за которым идут слова **то есть**. Найденное слово не подлежит замене, заменятся следующие за ним слова, конкретизировавшие поиск. **Простая, то есть - несложная. => Простая, то<есть - несложная.**

Еще круче: **все * *ло=всё ло | Все давно прошло. => Всё давно прошло. | Все вокруг потемнело. => Всё вокруг потемнело. | Все мне надоело. => Всё мне надоело.** И так далее... Чёртова уйма вариантов с омографом ВСЕ/ВСЁ решается одним правилом.

Заметим, что классический формат dic в этом отношении бессилён, ничего, кроме нудного добавления каждой из вариантных строк в словарь он предложить не может.

Разные мелочи вроде оптимального размера хеш-таблицы, разрешения коллизий и т.п. опускаю. Читайте библию от Д.Кнута "Искусство программирования" т.3 "Сортировка и поиск", глава 6, раздел 4 "Хеширование".

Скорость работы хеш-алгоритма практически на порядок выше, чем у "бесхитростной" версии метода прямого перебора. Конечно, его реализация требует от программиста больших усилий. Сравните: сколько слов я потратил на описание прямого перебора, а сколько на "хеш-алгоритм"?

<<

4. Нормальные герои, продолжение... или нет предела совершенству

*"Смотрите сюда... Это основная схема... Это просто, как дважды два.
Чистая случайность, что это до сих пор не было построено"*

Инженер Гарин

Во-первых, не откладывая в долгий ящик, устраним в нашем алгоритме одно слабое место. Вы, конечно, его заметили. Нет? Смотрите сюда. Допустим, в словаре есть 2 правила, расположенные в следующем порядке:

```
*ащенн*=ащённ
...
*враще*=враще<
```

Как будет выглядеть измененное слово "возвращенный"? При прямом переборе правил получим верный результат: **возвращённый**. Ибо правила с одинаковой длиной левых частей выполняются в порядке их следования в словаре. А вот как работает наш алгоритм. В хеш-таблице образуется *два* словарных гнезда:

```
*ащенн*|*ащенн*=ащённ
...
*враще*|*враще*=враще<
```

Их местоположение в хеш-таблице и порядок следования заранее неизвестны, да и не имеют значения. Выбор словарных гнезд происходит в том порядке, в каком генерируются ключи поиска процедурой Trunks! И как бы мы не переставляли местами сии 2 правила в словаре - итог будет один и тот же. Неутешительный. Потому что в "быстром алгоритме" в нужном порядке проверяются лишь правила, *сидящие в одном словарном гнезде*. А пока не сгенерирован следующий ключ, мы даже не знаем о существовании (или не существовании) очередного словарного гнезда. И не сможем понять, что текущая замена - преждевременна и делать ее не надо. Ибо найдется лучшая.

Поэтому, видоизменим алгоритм. Во-первых, генерируя для очередного слова список всех его огрызков, включаем лишь те огрызки, которые уже присутствуют в хеш-таблице. Отсюда: размер списка "подходящих огрызков" сократится в 10..20 раз. *Это - очень важное достижение*, как будет ясно из дальнейшего. Во-вторых, пройдясь по списку огрызков, выберем в хеш-таблице все подходящие словарные гнезда *и объединим их в общий список правил, потенциально подходящих к данному слову!* Отсортируем его в порядке убывания длин левых частей. И только после этого начнем применять правила из этого списка к данному слову (словосочетанию).

Во-вторых, сейчас мы сделаем то, до чего никто не додумался раньше. Как было сказано, в словарях формата DIC правила положено применять к тексту в порядке убывания длин их левых частей. Символ * в подсчет длины не входит. Если эта длина для 2-х правил - одинакова, то они применяются в порядке их следования в словаре.

ПРИМЕР 1

Правила:

```
*^*=
муха=слон
```

Текст: **м^у^ха**

Первым применяется правило с самой длинной левой частью: **муха=слон**. Оно - не подходит к тексту - замен нет; Затем выполняется правило ***^*=** - оно подходит, две замены. Просмотр словаря закончен, останов.

Измененный текст: **муха**

Всё верно. Но отчего-то кажется, что здесь что-то не так... А, вот оно что! Интуитивно мы ожидали, что правило ***^*=** сработает *первым*. Ведь оно и предназначено для предварительной подготовки текста. Увы - это противоречит давно устоявшемуся стандарту словарей DIC, по которому сначала проверяются *длинные* замены (что, кстати, вполне логично).

Но... Если к измененному тексту снова применить тот же словарь? Итак, всё сначала. Правило: **муха=слон** - теперь подходит! Одна замена. Правило: ***^*=** уже не подходит, свою службу сослужило раньше. Словарь исчерпан, останов.

Вторично измененный текст: **слон**.

Приходим к очевидному обобщению. Применять замены к тексту НЕ однократным проходом по словарю, а повторяя

этот процесс до тех пор, пока он приводит к изменению текста.

Однако, на этом пути нас подстерегает засада.

ПРИМЕР 2

Правило: ***чертя*=чертя<**

Текст: **очертя**

Измененный текст при многократном обращении к словарю:

**очертя<
очертя<<
очертя<<<**

и т.д. до морковкина заговенья.

Во избежание подобных казусов, должен вестись список уже примененных правил, и каждый раз применяться лишь те правила из словаря, которых еще нет в этом списке. Тем самым гарантируется конечность алгоритма.

Насколько будут отличаться тексты, измененные в традиционном и в только что предложенном порядке? Примерно на 0.1% в среднем. Так стоила ли игра свеч?

Полагаю, что да. Ведь текст (на любом языке) представляет собой т.н. "последовательность редких событий". Многие слова, несущие реальную смысловую нагрузку, встречаются в нем всего лишь несколько, а то и вовсе один раз! Оставаясь при том важными для понимания текста. Поэтому, улучшение алгоритма словарных замен, касающееся даже небольшой доли слов в тексте, может оказаться полезным.

ПРИМЕР 3

Словарь: Michelangelo.dic

Текст: **Devochka Nina - korr. CNN**

Измененный текст: **де<вочка нина - корреспонде<нт си энэ<нн**

Примененные правила:

1-й проход: а) раскрыта аббревиатура; б) текст очищен от транслита

спп=си энэ<нн

***ch*=ч | *a*=а | *d*=д | *e*=е | *i*=и | *k*=к | *n*=н | *o*=о | *r*=р | *v*=в**

2-й проход: а) применено правило постановки ударения; б) раскрыто условное сокращение

девочк*=де<вочк

***кorr.=корреспондент**

3-й проход - применено правило постановки ударения

корреспондент*=корреспонде<нт

Итак:

- 1) Проверка ведется по словарю в целом;
- 2) Правила применяются в порядке убывания длин их левых частей. Управляющие символы * и \$ в подсчет длины не входят;
- 3) Правила с одинаковой длиной левых частей применяются в том порядке, в каком стоят в словаре;
- 4) Звездочки могут находиться не только на концах, но и *внутри* левых частей правил;
- 5) В отличие от традиционной практики, правила применяются к тексту не однократной проверкой по словарю, *а до тех пор, пока возможно*.

Возвращение к истокам. "Умный" алгоритм, обеспечивающий наиболее полное использование словаря - это хорошо. Но, бывают случаи, когда "сильно хорошо - тоже плохо". Что если нужны замены в тексте *в строго определенном порядке*?

Поэтому, в Demagog доступен и алгоритм словарных замен прямым перебором правил по словарю. Для этого достаточно снять галочку в "Сервис - Общие настройки... - Чтение - Хешировать dic-словари".

Особенности реализации. Чтобы классический алгоритм прямого перебора работал приемлемо-быстро, видоизменим его. Будем применять перебор по списку правил не ко всему тексту сразу, *а к его небольшим фрагментам, "склеивая" результаты воедино*.

Дополнительную прибавку скорости дает следующий технический прием. Когда требуется замена с учетом и без учета регистра, то не надо поддаваться инстинктивному желанию написать что-то вроде:

if CaseSensitive

then k := Pos(FindText, S, k)

else k := Pos(Lower(FindText), Lower(S), k);

Выполняемое в цикле поиска приведение длинной строки S к нижнему регистру будет заметно тормозить дело. Лучше всего перед началом цикла создать копию S0 := Lower(S). После чего в цикле поиска/замены все действия выполнять сразу над S, и над ее копией.

Если в dic-словаре слишком много правил со звездочками... Представим себе словарь примерно такой структуры:

ПРИМЕР 4

***, а баски=, а бАски**

...

***, ядра=, Ядра,**

Как видим, все правила начинаются с одного и того же паттерна: *****, Пусть для применения правил выбран в Настройках хеш-алгоритм. Но, именно по первому паттерну, правила размещаются во временной словарной хеш-таблице.

Т.о. все правила из этого словаря попадут в одну и ту же строку хеш-таблицы! И разместятся там дружной толпой, одно за другим. И, вместо того, чтобы мгновенно возвращать по хеш-коду нужное правило, работа алгоритма сведется к нудному перебору всех правил в одной строке хеш-таблицы, в поисках подходящего. Учитывая дополнительные затраты времени на создание хеш-таблицы (и прочие сопутствующие действия), приходим к выводу, что в данном случае *хеш-алгоритм не будет эффективен*.

Рекомендация: Для *dic*-словарей, содержащих, в основном, правила со звездочками - лучше применять *алгоритм прямого перебора*. Для этого снять галочку в "Сервис - Общие настройки... - Хешировать *dic*-словари".

Неоднозначность *dic*-правил с отдельно стоящими звездочками. Если в шаблоне поиска есть отдельно стоящие звездочки, то программа предполагает, что каждому слову шаблона соответствует слово замены из правой части правила: **все * *ло=всё ло**

Если однозначное соответствие левой и правой частей отсутствует, то результат применения "звездного правила" будет далек от ожидаемого: **не о чем * * * *вать=нЕочем вать**

Не о чем нам с вами разговаривать! --> нЕочем вать нам с вами разговари! 🙄

Решение:

не о чем * * * *вать=нЕочем ~ ~ вать

~ =

Не о чем нам с вами разговаривать! --> нЕочем нам с вами разговаривать! 😊

Полный вариант хеш-алгоритма на языке Python 3.10 приведен [здесь](#). Он в 2 с лишним раза медленней, чем реализация на Delphi. Зато в питон-скрипте структура алгоритма очень ясно видна.

<<

5. Свежий взгляд

Поила *старая старушка*
Меня *молочным молоком*,
И я *глотки глотал* из кружки,
Языча губы *языком*.

Затем буханку с *хлебным хлебом*
И *маслом масляным* жевал,
И под *небесным синим небом*
Про это *рифмы рифмовал*.

Смеялась бабка, как чумная,
Потом *сказала*, щуря *глаз*:
"Как для издательства - не знаю,
А для пародий - в самый раз!"

Александр Матюшкин-Герке

В 1999 г. Д.Кирсанов написал утилиту Fresh Eye для поиска в тексте фонетически и морфологически схожих слов, стоящих "слишком близко" (<http://www.kirsanov.com/fresheye>).

Наткнувшись случайно на нее в Интернете, я решил посмотреть, как это все работает. Начал с очевидного:

Карл украл у Клары кораллы, Клара украла у Карла кларнет.

Как ни менял в настройках т.н. "порог чувствительности" - ответ получался один: проверено 1, найдено 0. А в стихах про старушку и поэта (при пороге чувствительности, принятом по умолчанию), отыскалось лишь 2 повтора: "маслом масляным" и "рифмы рифмовал". Ну и пусть... Одобрения заслуживает сама идея анализа текста на повторы, за каковую и спасибо.

Первым делом, необходим метод приблизительного сравнения слов. Можем же мы прочесть следующий текст:

По рзельуаттам илссеовадний одонго анлигйсокго унвиертисета, не иеemt занчнения, вкокам пряокде рсапожолены бквыы в солве. Галвоне, чотбы преавя и пслоендяя бквыы блыи на мсете. Осатълыне бквыы мгоут селдовтаъ в плоонм беспорядке, все-рвано ткест читаитсея без побрелм. Пичрионий эгото являтеся то, что мы не чаиатем кдаужю бкву по отдльенотси, а все солво цликеем.

Попробуем формализовать интуитивный процесс, благодаря которому мы поняли, что здесь написано. Определим коэффициент "похожести" двух слов, подсчитав, сколько символов из первого слова содержаться во втором. Но такой алгоритм не способен различать анаграммы и перевертыши. Например, слова "колесо", "оселок", "окосел" будут восприняты, как одинаковые.

Усложним процедуру, добавив проверку на присутствие 2-буквенных подстрок первого слова во втором. Потом 3-х буквенных... не пора ли остановиться? Средняя длина слова в русском языке - 6 букв. Хотите верить, хотите нет, но перебором 3-х буквенных подстрок (ровно половина от 6) можно ограничиться. Число успешных поисков поделим на общее число подстрок - это и будет показатель сходства.

Примеры:

КОЛЕСО, ОКОСЕЛ. По отдельным буквам 6 попаданий из 6. 2-х буквенные: КО, ОЛ, ЛЕ, ЕС, СО - 1 из 5. 3-х буквенные: КОЛ, ОЛЕ, ЛЕС, ЕСО - 0 из 4. Результат: $(6+1+0)/(6+5+4) = 0.467$.

АРБУЗ, ТАРАКАН. 2 из 5; 1 из 4; 0 из 3. Результат: $3/12 = 0.250$.

ТАРАКАН, АРБУЗ. 4 из 7; 1 из 6; 0 из 5. Результат: $5/18 = 0.278$.

Видим, что "сходство", вообще говоря, не коммутативно. Поменяв порядок сравнения, можно получить иной результат. Чтобы побороть такой эффект, будем вычислять "коэффициент сходства", усредненный по обоим словам. Общее число совпадающих подстрок разделим на общее число подстрок: $(3+5)/(12+18) = 0.267$

Этот, т.н. "алгоритм нечеткого сравнения строк" предложен в 1998 г. Владимиром Кива. На практике оказывается, что слова с уровнем сходства 0.35..0.50 зрительно и по звучанию уже весьма похожи!

Конечно, повторы в художественных текстах - совсем не обязательно признак неопытности автора. Зачастую они приносят в произведение дополнительную смысловую и эмоциональную составляющую. Поэтому, не станем объявлять графоманом великого писателя земли русской.

"В это время в гостиную вошло **новое лицо**. **Новое лицо** это был молодой **князь** Андрей Болконский, муж **маленькой княгини**. **Князь** Болконский был небольшого роста, весьма красивый молодой человек с определенными и сухими чертами. Все в **его** фигуре, начиная от **усталого**, скучающего взгляда до **тихого мерного шага**, представляло самую резкую противоположность с его **маленькою, оживленною женой**. Ему, видимо, все бывшие в гостинной не только были знакомы, но уж надоели ему так, что и смотреть на них и слушать их ему было очень **скучно**. Из всех же **прискучивших** ему **лиц, лицо** его хорошенькой жены, казалось, больше всех ему надоело..."

<<

6. "Глокая куздра", "четыре чetyрки" и питон

*Вспыхает небо, разбужая ветер,
Проснувший гомон птичьих голосов.
Проклиная все на белом свете,
Я вновь бежу в нетоптанность лесов.*

*Шуршат зверушки, выбегнув навстречу,
Приветливыми лапками маша -
Я среди тут пробуду целый вечер,
Бессмертные творения **пиша**.*

*Но, выползнув на миг из тины зыбкой,
Болотная **зеленовая** тварь
Советает мне с заботливой улыбкой
БОЛЬШОЙ ОРФОГРАФИЧЕСКИЙ СЛОВАРЬ.*

Александр Матюшкин-Герке

*Пусть безумная идея -
Не рубите с горяча.
Вызывайте нас скорее
Через гада главврача!*

*С уважением... Дата. Подпись.
Отвечайте нам, а то,
Если вы не ответите,
Мы напишем... в "Спортлото"!*

В.Высоцкий "Письмо в редакцию"

Demagog умеет говорить и анализировать текст на повторы и созвучия. А хорошо бы в придачу иметь (пусть даже простейший) контроль орфографии. Ведь так нередки в текстах АшиПки и оЧеПятки...

Разумеется, можно подключаться к **MS Word** средствами **Ole Automation**. Уж в Ворде-то проверка орфографии реализована, так пусть он и делает всю работу за нас. Правда, возникнут сложности с разными версиями Ворда. Что 2003-му здорово, то 97-му - смерть... Да и странно это - превращать изящную в своей компактности программу в заурядный придаток при некоем монстре. Есть ли другие пути?

Одна идея, словно призрак коммунизма, бродит среди лингвистически настроенных умов, начиная с середины прошлого века. Только-только появились первые ЭВМ... Это означало: электронная вычислительная машина, слово "компьютер" еще не придумали, но придумали "кибернетику" - буржуазную лженауку для запугивания рабочего класса 😊. Тут-то кое-кого и осенило.

Суть гениального (не знаю, нужны ли в этом слове кавычки) озарения такова. Количество 2-буквенных сочетаний в русском алфавите: $33 \times 33 = 1089$. А допустимых из них: раз-два и обчелся. Можете самостоятельно составить квадратную табличку и вписать в нее все "разрешенные" комбинации. Простейший алгоритм, считай, готов.

Безумие этой идеи в том, что минимальный набор проверочных комбинаций охватывает все потенциально возможные слова русского языка! Становится ненужным огромный словарь-тезаурис, где каждое слово надо указывать во всех падежах, наклонениях, спряжениях и т.п. (Ведь даже тезаурус системы Ispell на 850 тыс. словоформ, на практике оказывается не вполне достаточным).

Но... (спускаясь с небес на землю), мы скоро заметим, что из 2-буквенных комбинаций можно понаделать еще больше невозможных, несуществующих слов. Все они, естественно, были бы при проверке признаны "правильными". **Глокая куздра штеко будланула бокра и курдячит бокренка** - здесь только допустимые пары букв!

Очевидно, необходимо ужесточить схему, проверяя на допустимость не 2-х буквенные, а 3-буквенные сочетания. Тогда для составления проверочного списка придется прошерстить уже $33 \times 33 \times 33 = 35937$ троек в поисках правильных. Еще более надежным решением будет проверка "четырок". Например, в слове "демагог" - четыре чetyрки: дема/емаг/маго/агог. Все они - допустимые. Из общего числа в 1185921 - таких найдется тысяч 35-40. Проверая все чetyрки заданного слова на допустимость, делаем вывод о верном или неверном его написании. Все упомянутые в Интернета изыски по *бессловарной проверке орфографии* на этом исчерпываются.

Поразмыслив, можно найти примеры, когда и такая схема слишком уж всеобъемлюща. **Блинны**. Это слово не опознается, как ошибочное, т.к. 4-ка **линн** - допустимая. (**Длинный**). **Песьмо** - из той же оперы - **песь** - **спесь** - **песьими**. Надо что-то делать...

Попробуем перейти к проверке 5-ти буквенных сочетаний. Весело, дружно, хватаем бревнышко и несем... Что нам

попытки перейти к проверке в три секунды с помощью *ссылки*, *ссылки*, *ссылки* *ссылки* и *ссылки*. Не стоит среди 39135399 (39 миллионов с хвостиком) вариантов отыскать "правильные пятерки"?

Я использовал простенький скрипт на Python для обработки текста, в правильности которого был достаточно уверен. С помощью скрипта обрабатывался *очень* большой текст. Это - основное требование к тексту, чтобы он был громадным. Во-вторых, он не должен содержать знаков дефиса для переноса строк. В-третьих, он должен быть общехудожественным, а не специализированным.

Я составлял "большой текст" по частям, "склеивая" из отдельных файлов. В одном из них потребовалось удалить все знаки переноса - здесь помог Demagog. В итоговый bigtext.txt, размером 15 Мб, вошло с дюжину романов. Названий не привожу, произведения отбирались не по величию авторов, а по величине текста. Так быстрее получался должный объем "словесного материала". Разумеется, я пытался придерживаться принципа "общехудожественности". Никаких физико-математико-философских работ с зауменно-языколомной терминологией! Литературные произведения только вменяемых авторов... Мы же собираемся проверять орфографию *русского* языка.

Помимо этого, в тексте изначально надо удалить *все слова, написанные большими буквами, или начинающиеся с таковой*. Тем самым отмечаются имена собственные и географические названия. Жаль, конечно, казака Дормидонто Семижопенко из ст. Старонижнемухосранской, но зато избавляемся от всяческих **Хэххххрх Рхрч, Жугдэрдэмидийн Гуррагча, Ыгыатта, Таллинн, Кыргызстан** и т.д. и т.п. и пр. и пр. Убирать надо также слова с цифрами, нерусскими буквами и написанные через дефис.

Недопустимы и слова, содержащие 3 одинаковых буквы подряд, за исключением фрагмента "ошеее". Слова со следующими друг за другом парами одинаковых букв также не надо брать во внимание во внимание, если только это не пары: еенн, ллее, ллии, ллоо, ннее, ннюю, нняя, ссее, ссии. Тогда не страшны будут разные **Уурраааа!! Аррггх-бруумм! Тра-татаа!!** и тому подобные гремящие комбинации, которыми некоторые авторы "оживляют" повествование. Шутки шутками, но удастся избавиться от изрядного количества мусора в обрабатываемом тексте.

Насколько полон полученный список "пятерок"? Вот как менялся размер списка с ростом объема текста:

| Объем текста, мб | Количество найденных "пятерок", тыс. |
|------------------|--------------------------------------|
| 0 | 0 |
| 3.2 | 48.0 |
| 6.8 | 77.1 |
| 10.9 | 88.2 |
| 14.2 | 94.8 |
| 15.0 | 99.5 |
| ... | ... |

Каждая очередная порция текста добавляла все меньше и меньше нового. Вдалеке просматривается некий предел? Не обязательно. Возможно, количество пятерок растет, как *логарифм* объема текста. Во всяком случае, чем дальше, тем рост всё более медленный.

Пополнять bigtext.txt новыми шедеврами мне надоело, да и обработка его скриптом стала занимать аж целых 3 минуты - что же будет дальше? Решил пойти на хитрость. Нашел в Интернете частотный словарь русского языка на 69307 наиболее употребительных словоформ и скормил скрипту. Вышло, круглым счетом, 60 тыс. пятерок - так сказать, необходимый минимум. Добавил сей чудный словарь в bigtext.txt... На выходе - 103 тыс. - прирост совсем уже невеликий.

Тот же фокус можно проделать с тезаурусом системы Ispell. Из 850 тыс. наивозможнейших словоформ выходит 150 тыс. пятерок. Но тезаурусы Ispell составляются автоматически по принципу добавления к корням слов всех возможных приставок и окончаний, и содержат много маловероятных, практически неупотребляемых вариантов. **Подучилось** - нечто среднего рода повысило свои знания в некоторой области. А ведь, скорее всего, "подучилось" - это неверное написание слова "получилось"! Вот так: сильно хорошо - тоже плохо. От чрезмерного расширения списка эффективность отлова ошибок и опечаток начинает снижаться! Практика показывает, что 100 тыс. пятерок, полученных обработкой текстов общепотребительной лексики - вполне достаточно.

В полученной орфотаблице для каждой пятерки указывается также ее *код*. 100х - пятерка встречается только в начале слова; 010х - только в середине; 001х - только в конце. Возможные комбинации: 110х, 011х, 101х, 111х. х = 1 для буквосочетаний, которые могут быть самостоятельными словами и 0 в противном случае. Очевидно, что слова короче 5 букв имеют код: 0001.

Эти 4-значные коды логичнее записывать в виде однозначного 16-ричного числа. Размер файла орфотаблицы тем самым уменьшается на треть.

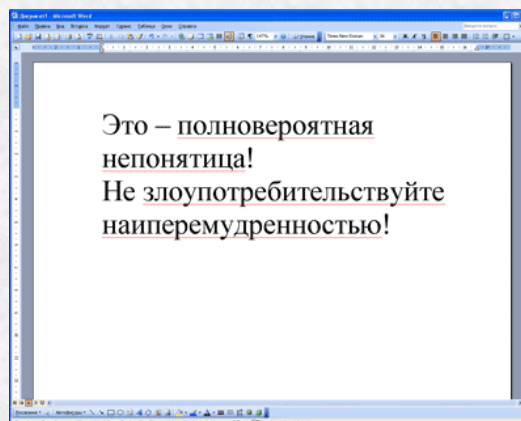
Посмотрим, как это работает. **Красавчег** = краса / расав / савче / авчег. Найдено в орфотаблице: краса 1111. 5-ка *может* находится в начале слова; расав 0100 - только в середине, она там и находится; савче - НЕТ В ТАБЛИЦЕ. ОШИБКА. **Нарей** мне стакан воды! В таблице есть: нарей 0110 - может находится только в середине или конце слова (канарейка, фонарей). ОШИБКА.

Нажатием клавиши F4 проверяется выделенный текст. При формировании орфотаблицы я старался не пропускать в обработку составные слова, потому что на "стыках" входящих в них "простых" слов могут получаться нехарактерные, редкие буквосочетания, которые лишь зря замусорят таблицу. Но составное слово можно проверить, по очереди выделяя отдельные его части; или разделив их пробелами или дефисами и, затем, выделив все целиком.

ПРЕДУПРЕЖДЕНИЕ для энтузиастов, которым захочется проверять 6-ти (и более) буквенные сочетания. Проверка станет работать хуже и чем дальше, тем хуже 😊. Все потому, что орфотаблица начнет постепенно превращаться в тезаурус, разбухать в размерах, ведь в нее заведомо потребуется включать все словоформы не длиннее n, где n = 6, 7, 8...

Кстати, n идущих подряд букв в слове называются *n-граммой*, этим термином и будем дальше пользоваться. В нашем случае n = 5.

Еще раз подчеркну *принципиальное отличие n-граммного метода проверки орфографии от словарного*. Например, в bigtext.txt, использованном для построения орфотаблицы, и в помине не было вот этих слов, которые MS Word в панике подчеркнул красным:



Интуитивно эти слова вполне понятны и кажутся написанными правильно! В определенном смысле это так и есть. Все содержащиеся в них 5-граммы, получены обработкой других слов из bigtext.txt и являются допустимыми. И для n-граммного метода вышеприведенная фраза - *верна*. Но попробуйте написать: **полнавироятная, нипонятица, злоупотребительстуйте, наиперемурденностью...** Эти слова (опять же, в полном согласии с нашей интуицией!) будут отбракованы.

Создается впечатление, что n-граммному методу присущ своеобразный "интеллект". Конечно, даже при n=5 он пропускает, как якобы верные, несуществующие слова. Но... они выглядят довольно естественно, укладываются в рамки статистических закономерностей языка.

В общем и целом, метод уверенно различает тексты, написанные с ошибками и без.

Стихи А.Матюшкина-Герке в эпиграфе этой главы расцвечены **красным** - слова которые и MS Word и Demagog дружно объявили ошибочными и **синим** - слова, которые только Word счел неправильными. Вполне удовлетворительно; как и ожидалось, орфотаблица из 105 тыс. элементов - достаточно полна.

Это - обычный текстовый файл, даже не отсортированный по алфавиту, ибо для Demagog порядок следования элементов в орфотаблице безразличен.

Содержимое орфотаблицы можно изменять. Но не надо усердствовать, засоряя ее именами собственными и географическими названиями. Они, как правило, не являются осмысленными словами и не подчиняются языковым законам. (**Тверь-универсалбанк**. Каково?)

А удаляя какую-нибудь "нехорошую" словоформу, необходимо помнить, что при этом удаляются составляющие ее 5-граммы. В их числе могут оказаться и нужные, правильные, порожденные другими, вполне нормальными словами. Чтобы "возместить потери", придется некоторые слова вносить в орфотаблицу заново. Отсюда правило: если необходимо одно добавить, а другое удалить, то сперва надо выполнить удаление!

Когда выполнена проверка орфографии (хоть маленького фрагмента, хоть всего текста - это занимает секунды, то в окне "Статистика" появится список найденных ошибочных слов, отсортированный по алфавиту. При этом слова, начинающиеся с большой буквы, помещаются в начале списка и отсортированы отдельно.

Кстати, еще о подмеченной нами "разумности" этого необычного, статистического по своей природе алгоритма. Писатель А.И.Солженицын составил т.н. "Русский словарь языкового расширения" (М. Русский путь, 2000), о котором многие лингвисты отозвались крайне отрицательно. Вот короткий рассказик, содержащий почти только одни слова из "Расширения".

Растопыря, или необходимая баба

Ерыжливый дурносоп верстан, достодолжный жегнуть шершавку, любонеистово айлил жиротопное шурьё. Зябкоподжимчивый валява остробучил, жубря: "Хунды-мунды, вахлюй! Отрезно ты фефёлу дочул, иззаплаченный дурандай!" "Да, жемнул я мормотень! - отжегнул дурносоп верстан, - "а тебе вот маламзя с расщепырей!" "Да, ить здесь одна жирным-жирнешенька шеврюжка!" - верстанулса прощепырник. "А ты чо выхайлился, захухряев оторвяжник?" - утомчилс зябкоподжимчивый дурносоп. "Эвось! - защепырил прожубрястый валявка, - "я то - чужеры! А ёна ведь неутомчивая жемжурка. Коли ей баларыст зажирнить в шабры, так расщепыритс захухрястой профефёлой!" баларыст зажирнить в шабры, так расщепыритс захухрястой профефёлой!" захухряй прожемнул поконце и ущепырил растопырю.

Можете сами скормить сей текст Demagog, выделить все, затем F4 и полюбоваться результатом. Никаких других доказательств, полагаю, не нужно. Ладно, там MS Word... Если уж и Demagog полностью отвергает *это*, то "слова" из "Расширения" и, в самом деле, не отвечают строю русского языка.

<<

7. "Говорит и показывает..."

Интервью с лидером Партии пофигистов.

- Почему ваша партия так называется?
- Потому что нам все пофиг.
- И проблемы экологии?
- Пофиг.

- И экономический кризис?
- Пофиг.
- А... так вы и к деньгам равнодушны?
- Э... нет! Деньги - это святое!
- А как же ваши принципы?
- Пофиг.

Старый анекдот.

Похожая история вышла и с моими принципами. Нет-нет, дело не во внезапном приступе жадности; Demagog как был, так и останется бесплатной программой. Но, в некотором смысле, мои убеждения оказались на поверку такими же гибкими, как взгляды некоего партийного босса. "Если без чего-то можно обойтись, то оно в программу включено не будет" - до сих пор я строго придерживался этого правила.

Голосовые движки понимают только обычный текст, отнюдь не документы MS Word или электронные книги в формате Fiction book. После преобразования оных в просто текст, все картинки, если таковые в исходных документах имелись, будут потеряны. "Это есть факт", всем известный и давно привычный. Жалко, но как-нибудь обойдемся.

Часто ли в современной "бумажной" книжке увидишь хоть одну иллюстрацию? Лишние расходы и хлопоты издателям ни к чему. "Пипл схавает" и так.

Но, все же... все же... Делать ничего не будем, а просто так посидим, подумаем. За компьютером. И Delphi запустим, по привычке.

Лишь на первый взгляд извлечение рисунков из doc, docx, rtf-файла представляется неразрешимой загадкой. Что будет, если средствами OLE Automation заставить MS Word сохранить некий документ, пусть это будет "Алиса в стране чудес.doc", как веб-страницу "Алиса в стране чудес.htm", и сразу ее удалить?

Правильно. Приз в студию. Останется папка с именем "Алиса в стране чудес.files", содержащая все рисунки из первоначального документа. (О том, что для ранних версий Word этот фокус не работает, культурно умолчим).

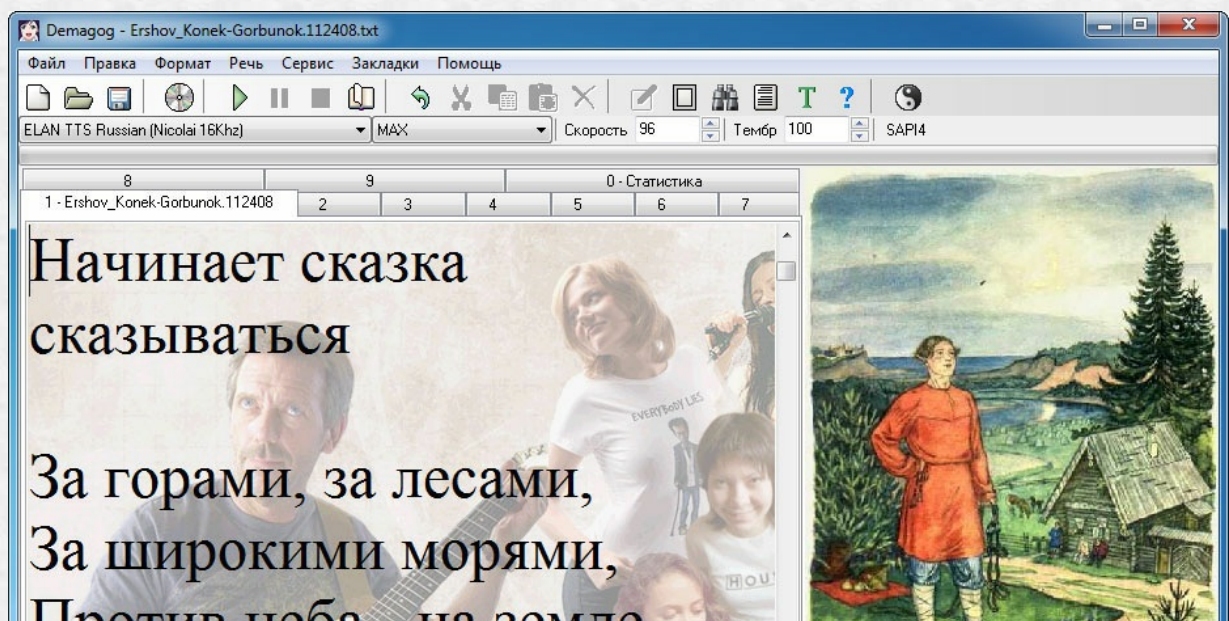
А что если в Demagog загрузить *именно веб-страницу*, когда-то выкачанную нами из Сети? Вспомним, что при сохранении веб-страницы, каталог "имя-страницы.files" создается автоматически. Правда, не всегда - иные веб-страницы порождают каталоги с именем: "имя-веб-страницы_files". Программа должна проверять оба варианта названия "каталога-галереи".

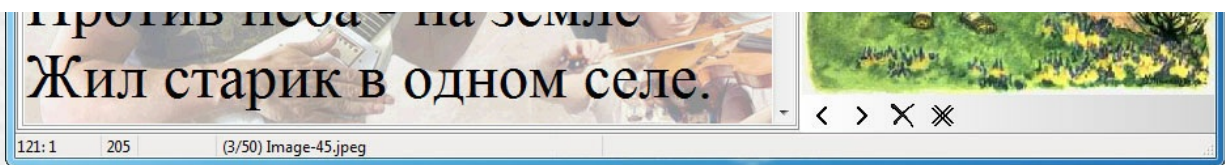
Итак, с вордовскими текстами и веб-страницами разобрались. Займемся электронными книгами формата fb2 - заурядными XML-документами. Картинки в них заданы блоками т.н. кода Base64. Например: <binary id="faraon.jpg" content-type="image/jpg">...код...</binary>. Вырезать эдакие подстроки и поместить их в список TStringList, ясное дело, раз плюнуть. Исходник функции Base64decode() возьмем в Сети.

Запишем декодированные бинарные строки в файлы с соответствующими расширениями, поместив их, как нам уже привычно, в папку "что-то там такое.files". Видим, что полученные jpg и bmp-файлы компонентом Image отображаются правильно! А вот с форматом png Delphi не работает принципиально. Ну, не было такого формата в эпоху зарождения и взлета Delphi. Но и здесь дела обстоят не так уж плохо. В Сети лежит-ждется, пока его скачают, компонент [TPNGImage](#). Все содержимое архива копируем в папку нашего проекта (естественно, без справки). В секцию uses проекта добавляем PngImage, и дело в шляпе. Теперь png-картинки грузятся без проблем, наравне с jpg и bmp. (Кстати, начиная с Delphi 2009 TPNGImage уже включен, как стандартный).

"Рисунки GIF забыли!" - возопил мой внутренний голос, но заткнулся, когда в секцию uses я дописал: *GifImage*. Больше ничего и не нужно. Ибо в Delphi 2007 уже имеется поддержка gif.

Осталось решить, где лучше отображать картинки. Проще всего - в специальной Галерее, справа от текста. Отступ Галереи от левого края в *процентах* задается в настройках. Если указать 50%, то тексту и картинкам достанется места поровну. По умолчанию: 65%. Для листания присобачим кнопки "взад-вперед" и, на всякий пожарный, "Удалить картинку" и "Удалить все". Дефолтные размеры окна у Demagog невелики и картинка будет похожа на большую почтовую марку, но это - дело поправимое. При изменении размеров окна программы, пропорционально поменяется размер картинки. Потихоньку, полегоньку... вот [такая](#) икебана:





- Эй-эй! А откуда взялся *фоновый рисунок под текстом?! -* спросите вы. Об этой мелочи как-нибудь потом.

<<

8. Чтение многоязычных текстов. Мультилингва

Выйду на поле в мятых трусах коричневых,
Знаю в футболе пару финтов гарринчевых¹.
На деревяшке выжгу тебя паяльником,
Скину рубашку, спрячусь под пододеяльником.

I love you baby.
I just believe in what you say.
Yes. It is table. And it was table yesterday.
I love you baby. Baby, you love my "имидж".
O-o. I am crazy. I can speak English!

Пародия на песню А.Серова "Ты меня любишь"

¹Гарринча - знаменитый бразильский футболист

Допустим, что пользователь желает прослушать текст на двух языках. Допустим, он изучает английский, и хочет утвердить в своей памяти слова пока еще малознакомого языка. Единственного, изучение которого окупается и стОит затраченных времени и денег. Языка международного общения.

Типичное решение: вставить в текст в нужных местах теги SAPI5 для смены голоса. Вот пример переключения на английский: **Привет, мир! <voice required="Name=Microsoft Zira Desktop"> Hello, World </voice>**. Выглядит довольно громоздко. Но это - работает. Разумеется, соответствующие русский и английский голоса должны быть установлены на компьютере.

В настоящее время в Demagog поддерживаются еще один - альтернативный способ чтения многоязычных текстов, также основанный на специальной разметке. Рассмотрим его подробнее.

Мультилингва. Режим чтения текста, написанного с использованием двух и более языков. Необходима предварительная разметка текста специальными *командами смены голоса*. Вручную, а для двуязычного русско-английского автоматически. Команда смены голоса имеет вид:

{{ИмяГолосовогоДвижка,Скорость,Тембр,Громкость}}

Пробелы внутри команды *недопустимы*. Имя голосового движка можно указывать не полностью, лишь бы оно было уникальным. Скорость и тембр - цифры в диапазоне: -10..10, громкость: 0..100. Стандартные значения: 0,0,100 можно не указывать. Тогда в двойных фигурных скобках будет записано только краткое имя голосового движка. Пример:

{{Irina}}Привет, мир! {{Zira}}Hello, World! {{Ludovico,,,80}}Saluton mondo! {{Zira}}Program "Demagog" is the speaking text editor!

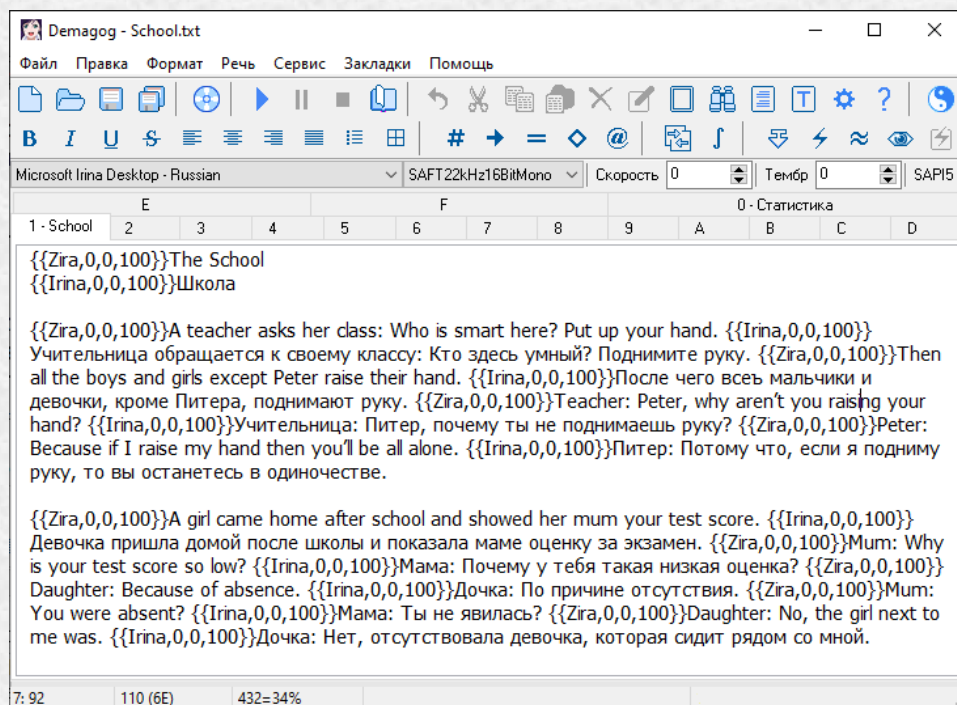
Таким образом, программа может читать текст, написанный на разных языках. Мультилингва работает только под SAPI5. Разумеется, мультилингву можно использовать и для чтения одного только русского текста - на разные голоса.

Наиболее часто встречающийся (для русскоязычных пользователей) вариант - это двуязычный, русско-английский текст. В меню "Речь - Мультилингва - Настроить" открывается окно настройки для чтения таких текстов. По умолчанию программа сама находит доступные в системе русский и английский голоса. При этом программа сокращает чрезмерно длинные названия голосов, удаляя из них цифры, поясняющие слова и знаки препинания. Алгоритм этих действий записан в файле ..\profiles\voices.re, и может быть откорректирован опытным пользователем, знакомым с языком "регулярных выражений".

Если в операционной системе установлено более одного русского/английского голоса, и автоматическая настройка выбрала не тот голос, который нужен пользователю, то соответствующая правка настроек производится вручную.

Если же, при автоматической настройке, названия русского или английского голоса остались пустыми, значит, программе не удалось найти соответствующего голоса, из числа установленных в системе.

Русско-английский текст, открытый в текущей вкладке, можно проверить на типичную ошибку - когда вместо русских букв в слове стоят английские с тем же начертанием. Или наоборот. Если программа обнаружит в тексте такие "кириллически-латинские" слова, то выведет их список во вкладку Статистики. Если же всё в порядке, то можно сделать автоматическую разметку текста нажатием кнопки "Вставить разметку". Увидим что-то вроде:



Теперь, чтобы размеченный текст читался на два голоса, включим в программе режим мультилингвы: "Речь - Мультилингва - Вкл / Выкл". Включение отобразится галочкой в меню и сообщением в строке состояния.

Примеры чтения русско-английского текста: [#1](#) (Win 10, Tatyana + Zira) [#2](#) (Win 10, Tatyana + Zira)

Примеры чтения многоязычных текстов: [#1](#) (Win 7, Maxim + Dmitri + Anna + Kyoko) [#2](#) (Win 10, Pavel + Nicolai + Maxim + Tatyana + Irina).

Запись многоязычного аудио можно делать и с помощью специального скрита. Тогда нам будет неважно, каковы настройки Мультилингвы в программе, и настроена ли она вообще. Ниже приведен пример текста и скрипта для автоматического получения двуязычного аудиофайла.

Demagog - Joe.dxt

Файл Правка Формат Речь Сервис Закладки Помощь

ELAN TTS Russian (Nicolai 16Khz) SAFT22kHz16BitMono Скорость 0 Тембр 0 SAPI5

| | | | | | | | | | |
|---------------------------------|---------------------|-------------|-----------------|------------|-------------|---|---|---|----------------|
| 7 - School | 8 - School2 | 9 - School3 | A | B | C | D | E | F | 0 - Статистика |
| 1 - Vseminij sledopit 1929 № 08 | 2 - SuperFilterTest | 3 - Joe | 4 - Путешествия | 5 - Пророк | 6 - Накалка | | | | |

Упражнение номер один, начали

Кто там, вдалеке? Who is there, far away? Who is there, far away?

Это - ковбой Неуловимый Джо. This is the cowboy Elusive Joe. This is the cowboy Elusive Joe.


Неужели никто не может поймать его? Can no one catch him? Can no one catch him?

Да, его никто не может поймать. Yes, no one can catch him. Yes, no one can catch him.

Но, почему? But why? But why?

Потому что он - никому не нужен! Because nobody needs him! Because nobody needs him!

Конец упражнения



7: 24 1087 (43F) 213=43% (1/1) Joe.jpg

```
-- ЗАПИСЬ РУССКО-АНГЛИЙСКОГО АУДИО ДЛЯ ТЕКСТА В АКТИВНОЙ ВКЛАДКЕ
-- Дано: русские фразы чередуются с их английским переводом; английский
-- перевод повторяется дважды с 2-х секундной паузой; перед произнесением
-- перевода необходима пауза 4 секунды - чтобы ученик успел предложить
-- свой вариант перевода, а затем услышал правильный

-- запомнить параметры первоначально активного голоса
vName, vRate, vPitch, vVolume = Voice()

k = WActive() -- номер активной вкладки

-- на время работы скрипта включить многоязычный режим
o = {}
o.Reading_Multilingva = true
Settings(o)

-- настройка голосов (скорость, тембр, громкость не указаны, по умолчанию 0,0,100)
ru = 'Pavel'
en = 'Zira'

-- временные словари для пауз (их можно создать где угодно, не обязательно в папке dic)
h = HomeFolder('dic')
SaveToFile('{''{{{(\\|?|!|+)=<silence msec="4000"/>$1}}}',h..'tmp_pause4.rex')
SaveToFile('{''{{{(\\|?|!|+)=<silence msec="2000"/>$1}}}',h..'tmp_pause2.rex')

-- привязать словари к голосам (если путь не указан, то подразумевается стандартная папка ..dic)
d = {}
d[Voice('Pavel')] = {'tmp_pause4.rex'}
d[Voice('Zira')] = {'tmp_pause2.rex'}

-- копию текста в активной вкладке разметить и поместить в Статистику
WMarkup(k,0,ru,en)

-- словарные замены во вкладке Статистика
WFilter(0,0,d)
```

```
-- запись двуязычного аудио по тексту в Статистике
-- т.к. измененный текст не был сохранен, то имя аудиофайла
-- генерируется автоматически из даты и времени)
WAudio(0)
```

```
-- удалить временные словари
os.remove(h..'tmp_pause4.rex')
os.remove(h..'tmp_pause2.rex')
```

```
-- восстановить активность первоначального голоса
SetVoice(vName,vRate,vPitch,vVolume)
```

В результате выполнения скрипта получено [аудио](#).

<<

9. REX-словари. Склонятор. Поиск в тексте

Demagog, кроме словарей типа DIC, о которых подробно рассказано в предыдущих главах сей повести, "понимает", начиная с версии 3.10.142, корректировочные словари на основе так называемых [регулярных выражений](#).

Для таких словарей я придумал расширение **.REX** - от слов **Regular Expressions**. Кроме того, Rex означает "король" - король словарей, самое эффективное средство для предварительной обработки текста. Если одновременно подключены словари DIC и REX, то словарь REX применяется к тексту первым. Могут быть одновременно подключены несколько словарей одного типа, тогда они применяются к тексту в алфавитном порядке их имен.

Задача 1. Заставить программу читать текст "по словам". Этого можно добиться, поставив в тексте точку после каждого слова. Сделать такое вручную для сколь-нибудь длинного текста - немыслимо.

Решение. Отдельное слово в тексте - это последовательность, состоящая из алфавитно-цифровых символов и знака подчеркивания (обозначаемых метасимволом \w). То есть:

`\w+`

где знак + указывает, что в слове должно быть *не менее* 1-го символа. Чтобы при поиске в тексте очередное слово запоминалось, добавим скобки:

`(\w+)`

и укажем, что найденное слово заменяется им же самим с точкой в конце:

`(\w+)=.$1.`

Здесь \$1 - это ссылка на запомненное значение скобочной группы номер 1, а она у нас всего одна и есть. Скобочные группы (...) принято называть *подвыражениями*. Demagog допускает до 15 подвыражений в одном правиле. Ссылки на подвыражения с номерами больше 9 надо записывать, заключая номер в фигурные скобки, например: \${10}

Задача 2. Читать каждое слово в тексте трижды.

Решение. `(\w+)=.$1 $1 $1`

Задача 3. Читать каждую строку текста трижды.

Решение. Предполагая, что каждая строка, кроме слов может содержать также пробелы, знаки препинания, кавычки, тире, круглые и квадратные скобки и знак апострофа, и заканчиваться символами возврата каретки и перевода строки, запишем: `([\w\x20\.,!/?\:\'\-\(\)\[\]\']+[r\n])=.$1$1$1`

Перечень символов положено заключать в квадратные скобки, вот так: [...]+. Ну, а \x20 - это, разумеется, 16-ричный код символа "пробел".

Задача 4. Прочсть английский текст русскоязычным движком.

Решение. Словарик [Lat2Cyr.dic](#) - это простейший "транслитер", переводящий латиницу в кириллический текст. Он - заведомо не полон и ждет энтузиастов, которые бы его усовершенствовали. Сейчас это - лишь "сухой остаток" от известного бесплатного словаря Michelangelo. Поскольку правила в Lat2Cyr.dic не содержат символов ударения, то он пригоден практически для любого русскоговорящего движка. This is a table -> Сзис ис э тэйбл!

Задача 5. Если пользователь не желает слушать английские вставки, если они ничего не дают ни уму его, ни сердцу, то не проще ли их вообще игнорировать?

Решение. Это сделает словарик KillLat.rex, состоящий всего из одной строчки:

`[\(\)]*[A-z]+[\(\)\.,!/?\:\'\-\(\)\[\]\']*=`

Из текста будет удалена любая последовательность латинских букв, числом не менее одной, за которой, возможно, следуют знаки препинания.

Еще одна задача. Составители корректировочных словарей часто сетуют на необходимость выписывать все варианты одного и того же слова. Приходится перебирать по очереди падежные окончания для мужского, женского, среднего рода. А кроме существительных... есть еще глаголы, причастия, деепричастия... Кошмар!

Решение. [Здесь](#) лежит словарик формата REX, который каждое слово в ед. числе и именительном падеже заменяет на список всех его возможных словоформ. Насчет "всех возможных" - это я прихвастнул, но результат действительно впечатляет. Это - "альфа версия" словаря, кое-какие словоформы он пропускает, а некоторые типы слов обрабатывает некорректно. Начало, однако, положено!

<<

10. Встроенный интерпретатор

К вопросу о целесообразности
использования клавишных духовых
инструментов священнослужителями
младшего и среднего звена на
внецерковных праздничных мероприятиях.

На хрена полу гармонь?

Пословица

Тема для диссертации

Demagog умеет выполнять вычисления по заранее заданным в текстовом файле командам. Такой текст, содержащий только команды и комментарии к ним, называется *скрипт*. Для выполнения скриптов в Demagog используется встроенный интерпретатор языка программирования [Lua](#) (версии 5.3).

Файлы скриптов должны иметь расширение .lua. Чтобы выполнить скрипт, загруженный в активное окно Демагога, нужно нажать F2. Такой режим полезен для отладки скрипта. (При этом допускается и расширение .txt для имени скрипта). Но наиболее практично выполнять скрипты, не загружая их в Demagog, а непосредственно из файла. Для этого файл скрипта должен находиться в папке Demagog\Tests_. Тогда его можно увидеть в меню "Выполнить скрипт - Из файла". Несколько примеров скриптов поставляются с дистрибутивом программы.

Могут спросить: а зачем оно надо? Ответ такой. Встроенный интерпретатор позволяет выполнять не только команды языка Lua, но и инициировать выполнение некоторых важных опций Демагога. Для этого интерпретатор пополнен несколькими "импортированными" функциями, отсутствующими в самом языке. Например:

```
ind = WActive() -- получить номер активного окна
-- обработать текст в активном окне указанным словарем
WFilter( ind, ind, HomeFolder('dic')..'Michel.dic')
WAudio(ind) -- записать аудио
ShowMessage('Работа закончена!')
```

Чуть более сложный скрипт:

```
-- Пример скрипта для программы Demagog
-- Запись аудио с выключением компьютера

cap = 'Выключить компьтер после записи аудио?'
itms = { ' ДА', ' НЕТ'}
a = Menu(cap,itms,2)
if a == 0 then goto HALT end
if a == 1 then
    ShowMessage('После записи аудио, компьтер будет выключен!')
end

-- выбрать документ
fname = OpenFileDialog()
if not fname then goto HALT end

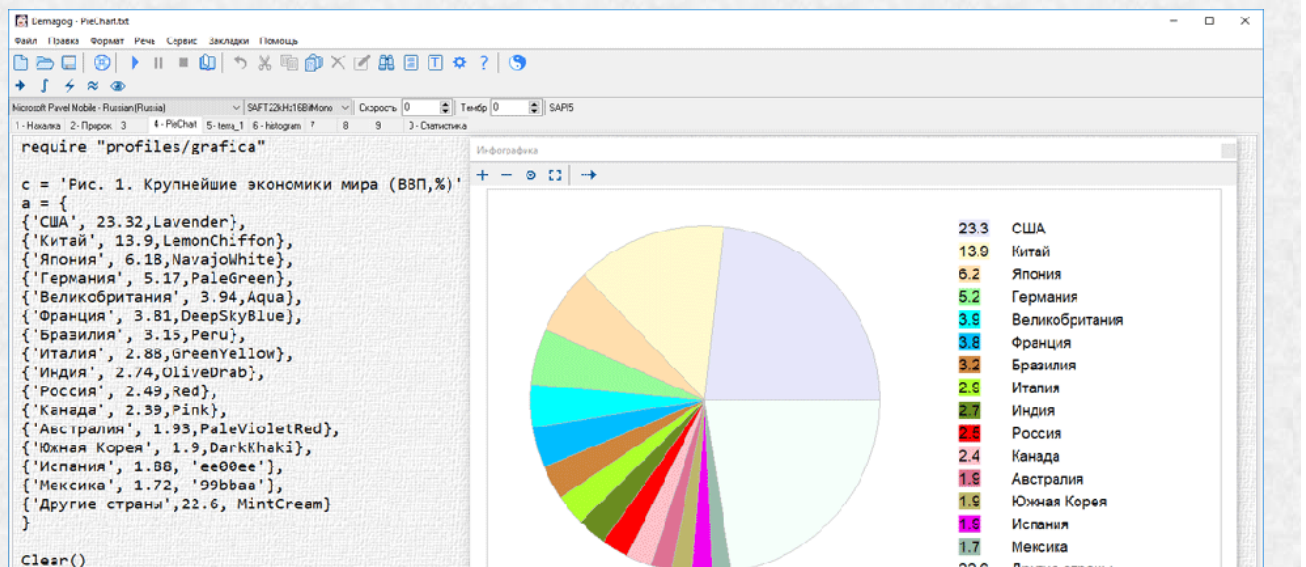
-- открыть документ в текущем окне и записать аудио
ind = WActive()
WOpen(ind,fname)
agree = WAudio(ind) -- agree - подтвердил ли пользователь папку назначения для аудио

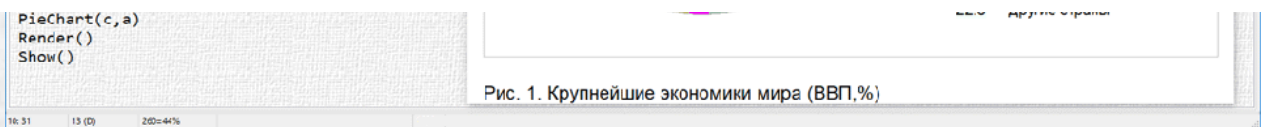
if a == 1 and agree then
    os.execute('shutdown /s') -- выключить компьютер
end

::HALT::
```

Это - простые примеры, но они наглядно демонстрируют возможность автоматизации рутинных действий пользователя. В заключение, прослушайте [фрагмент аудиозаписи](#). Она создана с помощью скрипта, написанного пользователем **balaamster**. Скрипт заставил две утилиты для обработки звуковых файлов выполняться совместно с Demagog, и под его управлением.

Начиная с версии 7.29.363, встроенный интерпретатор может выводить результаты своей работы не только в текстовой форме, но и в виде графиков и диаграмм. Полученные изображения можно сохранить в файлы формата png для дальнейшего использования.





Руководство пользователя: Пока что сверх-краткое (только на русском и английском) описание интерпретатора, включающее также полный перечень импортированных функций. "Сервис - Статистика - Выполнить скрипт - Шпаргалка".

Общие же мануалы по языку Lua в изобилии найдутся в Интернете. В настоящее время Lua широко используется в проектах, где нужен простой и мощный встраиваемый скриптовый язык.

<<

11. Вызов программы из командной строки

Все очень просто: **Demagog.exe имя_файла список_параметров**. Список параметров - необязателен. Параметры в списке разделяются пробелами и могут следовать в любом порядке. Их значения следующие:

| | |
|-----------|--|
| /m | - запустить программу свернутой в значок |
| /r | - открыть файл и начать чтение вслух |
| /s | - открыть файл и сохранить текст в виде аудиофайла |
| /c | - прочесть вслух текст из буфера обмена |
| /q | - закрыть программу после окончания чтения |

Например: **Demagog "C:\Tests\Некий текст.txt" /m /r /q**

<<

12. Подсветка ключевых слов в компоненте RichEdit

(Или снова о проверке орфографии)

Этот раздел посвящен вопросу, который рано или поздно, задает себе каждый, программирующий на Delphi. "У меня есть некий список слов, которые при отображении текста компонентом RichEdit должны выделяться цветом. КАК ЭТО СДЕЛАТЬ?!"

Например, хорошо бы подсвечивать в тексте орфографические ошибки. А то Demagog может лихо проверить хоть целую книжку, выдавши список ошибок и... ищи-свищи их по всему тексту. Неудобно. Выделение (по желанию пользователя) неправильных слов цветом было бы очень полезно.

Требуется решить проблему с минимальными усилиями, используя стандартные возможности компонента RichEdit. Решение очевидно: выделить нужное слово и через SelAttributes.Color поменять цвет выделенного. Затем снять выделение. И т.д. бегом по тексту, пока все нужные слова не будут найдены и раскрашены. Очень просто, не так ли?

Но... вот как комментируют этот алгоритм в сборнике [DRKB](#): Все способы подкраски синтаксиса реализованные через RichEdit грешат одним существенным недостатком - они реализованы через изменение атрибутов текста. И чем это грозит? А представьте себе что вы загрузили файл Дельфи, большой такой на пару мегабайт, например интерфейс от какого-то ActiveX от MS Word... и решили написать комментарий в начале файла, открываете скобку "(" и ... ждёте секунд 10, а то и минуту пока изменятся атрибуты у всего файла, затем закрываете скобку ")" и ждёте следующие пол минуты... Если же текст побольше, например вы загрузили какой-нибудь XML мегабайт на 50, то тогда после каждого нажатия клавиши у вас будет время выпить пивка и пройти уровень в Quake (желательно на другой машине, чтоб не тормозила)...

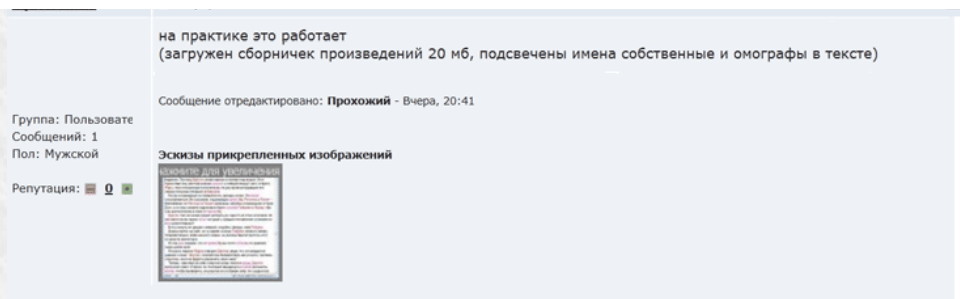
Далее автор комментария объявляет саму идею порочной, советует обрабатывать за один раз только фрагмент текста, видимый на экране, и изобрести собственную процедуру прорисовки раскрашенного текста в окне, что технически сложно, и вообще не изобретайте велосипед, а используйте вместо RichEdit сторонний компонент SynEdit.

Почесав в затылке и погуляв по разным форумам в поисках свежих идей, я высказал и свои скромные соображения. Дескать, приведенное решение - это не бог весть что, но чуток поправить и будет работать... И сразу получил суровую отповедь.

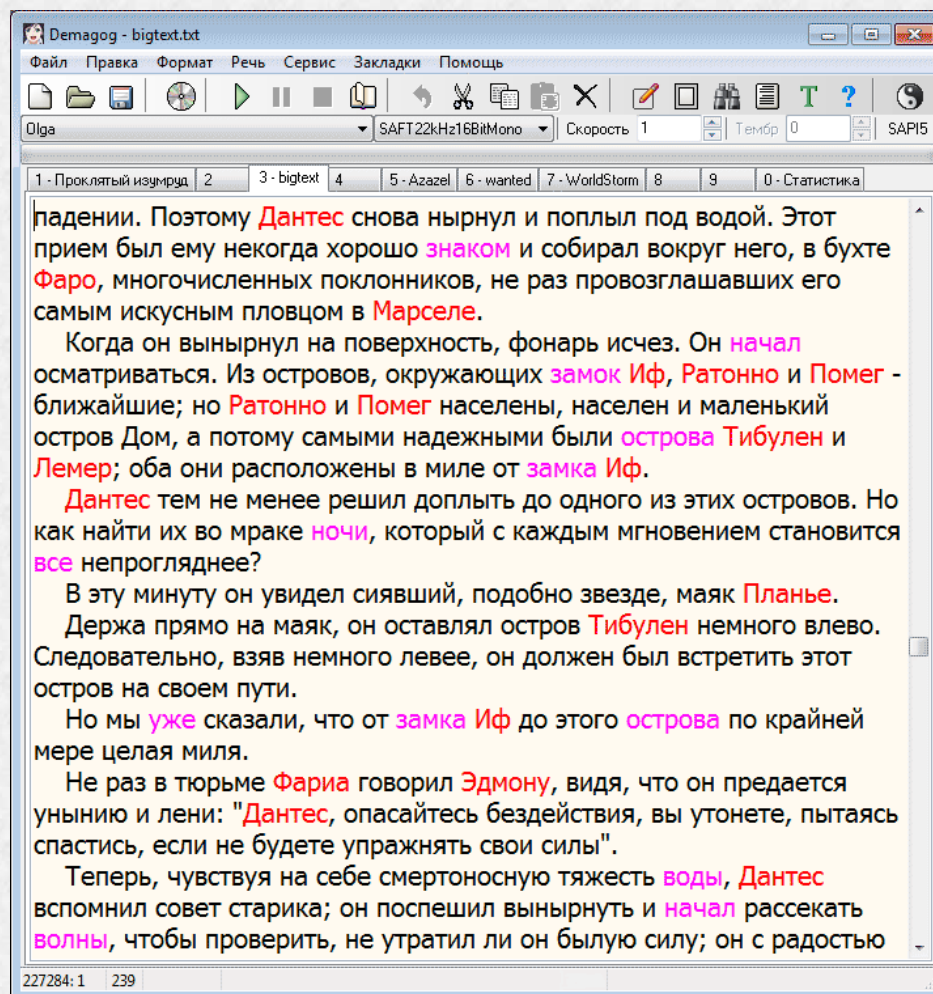
| | | |
|--|---|--------------|
| Прохожий | Вчера, 18:59 | Сообщение #5 |
| Гость | <p>Пляски с Селаттрибуес надо устраивать не по всему тексту, а только в той его части, что видна в окне - это совсем небольшой объем. Перед циклом "выделение - смена цвета" поставить ЛайнесБегинАпдейт, после - ЛайнесЭндАпдейт - тогда раскраска будет возникать в окне вся сразу, без мельтешения.</p> <p>Процедуру раскраски привязать к событию ОнКейАп для навигации по тексту. Но не ОнКейДаун, понятно, почему 😊</p> | |
| <div> <div>ЦИТАТА</div> <div>ОТВЕТИТЬ</div> </div> | | |
| IUnknown | Вчера, 19:04 | Сообщение #6 |
| <p>a.k.a. volvo877</p> | <p>Это теоретические измышления. Практическая реализация тут же покажет несостоятельность этих измышлений. Всё это хорошо "на бумаге", и если бы было так просто - то и вопросов бы не было.</p> | |

Каюсь, не удержался ответа:

| | | |
|----------|--------------|--------------|
| Прохожий | Вчера, 20:34 | Сообщение #7 |
|----------|--------------|--------------|



Скриншот в полном виде:



Поскольку орфотаблица составлялась без учета имен собственных и географических названий, то при проверке орфографии все такие имена и названия подсвечивались красным. Розовым подсвечены слова, имеющие различное ударение в зависимости от контекста - т.н. омографы.

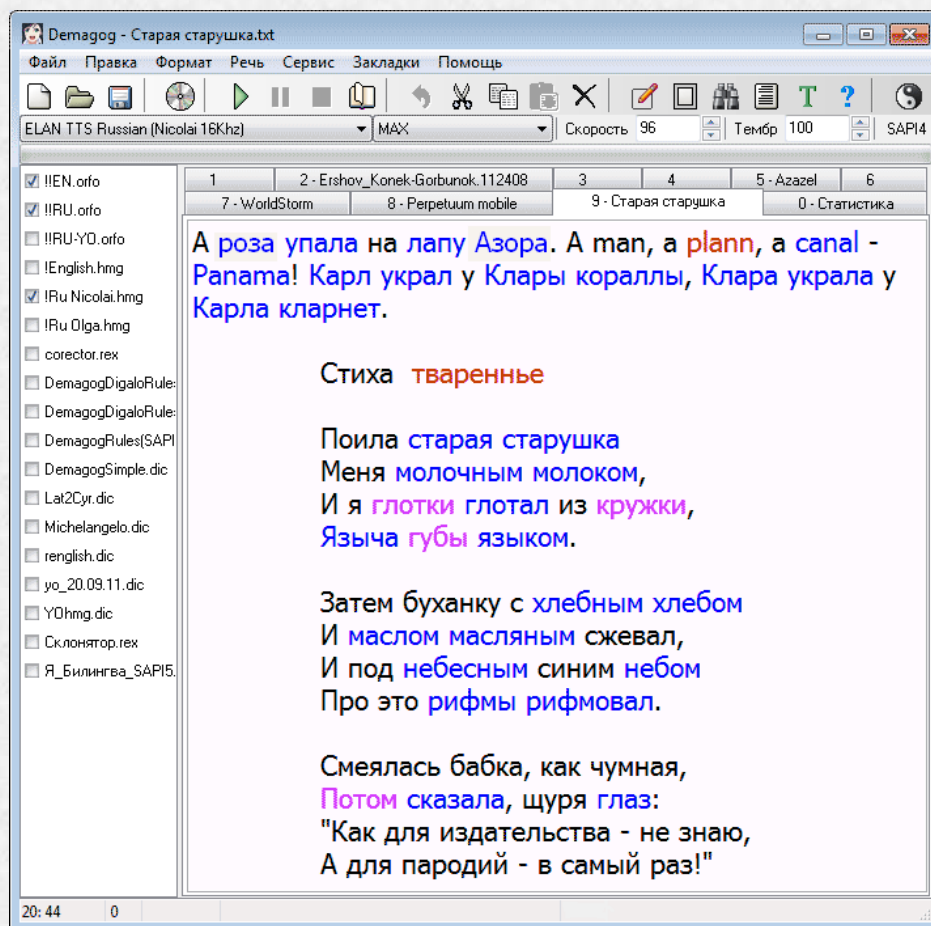
В программе используются словари омографов *.hmg - для различных голосовых движков. Они размещаются в подкаталоге disc рабочего каталога программы и видны в общем списке словарей. Их формат аналогичен принятому в программе "Балаболка". Рекомендуется давать словарям омографов имена, начинающиеся с символа !, тогда все они будут располагаться в начале списка словарей.

При подключенном словаре омографов в меню "Правка - Изменить выделенное" пользователю предлагаются для выделенного омографа варианты замен с различными ударениями. Например, при подключенном "!Ru Nicolai.hmg" двойной клик по слову "глаза" выделяет его, а нажатие правой клавиши мыши выдает меню с вариантами замен.

Файлы орфотаблицы помещаются в подкаталоге disc рабочего каталога программы и имеют расширение .orfo. Орфотаблиц может быть несколько, они видны в общем списке словарей. Может быть одновременно подключено несколько орфотаблиц, например одна для английского, другая для русского языка. Тем самым обеспечивается проверка орфографии в смешанном русско-английском тексте.

Учтите, что опции редактирования орфотаблицы доступны лишь тогда, когда подключена (отмечена галочкой) только одна орфотаблица!

Даже подсветка всего сразу: орфографии, омографов и похожих слов несколько не тормозит навигацию по тексту. Тем самым проблема быстрого поиска ошибок и несуразностей в тексте практически решена.



Кстати, тот критик с форума сразу заткнулся и больше не возникал 😊

P.S. А *настоящее* слабое место алгоритма, опубликованного в DRKB - совсем в другом! Он молчаливо предполагает, что всё, что вы видите в окне RichEdit - это некая строка текста `S := RichEdit.Text`. Стоит ее должным образом пропарсить, как мы получим набор позиций начала раскраски и длин окрашиваемых фрагментов, кои будем присваивать свойствам `RichEdit.SelStart` и `RichEdit.SelLen`. После чего изменим цвет очередного, выделенного таким образом фрагмента.

Так вот, всё это верно лишь для компонента RichEdit 1.0. В более старших версиях значение свойства `RichEdit.Text` *не совпадает с текстом, хранимым в памяти*. Удалены служебные символы разметки таблиц и списков, одинарные #13 заменены на #13#10 и т.д. и т.п. Замечание: в юникодных версиях Delphi используется RichEdit 2.0.

Поэтому, чтобы извлечь из RichEdit аутентичный текст для парсинга, необходимо использовать сообщение `EM_GETTEXTEX` с параметром `flags := GT_RAWTEXT`.

<<

13. Как добавить в программу новый язык интерфейса?

Выберем любой из имеющихся файлов языковых ресурсов в каталоге `..\languages`, например, `Russian.In`. Если вы не владеете русским, то возьмите `English.In`. Английский в наше время знают все.

Скопируем выбранный языковой файл, дав ему *английское* название испанского языка: `Spanish.In`. И откроем новый файл любым текстовым редактором. Да хоть самым Demagog.

Увидим (показаны только фрагменты):

```
...
[ACreate]
C=Новый
N=Новый|Создать новый документ

[AOpen]
C=Открыть...
N=Открыть...|Открыть документ

...

[lbRate]
C=__ Скорость__

[lbPitch]
C=__ Тембр__

...
```

```
[Msg_RulesWith]
C=Список замен по "%s". Правил: %d, применены %d раз
...
```

Наша задача: перевести на испанский всё, что идет после знаков = оставляя на месте знаки | и % если такие встречаются. Вместе со знаком % не трогаем латинские буквы, идущие вплотную за ним.

Знак нижнего подчеркивания _ на самом деле означает *один пробел* и предназначен для выравнивания некоторых заголовков, чтобы соответствующие элементы интерфейса располагались в правильном порядке.

Усердно потрудившись, получим что-то вроде:

```
...

[ACreate]
C=Nuevo
H=Nuevo|Crear un nuevo documento

[AOpen]
C=Abierto...
H=Abierto...|Abra el documento

...

[lbRate]
C=__Velocidad__

[lbPitch]
C=__Timbre__
...

[Msg_RulesWith]
C=Lista de sustitución basado "%s". Reglas: %d, aplicado %d veces
...
```

Сохраним файл Spanish.In в том же каталоге ..\languages в кодировке Unicode.

При новом запуске Demagog в меню "Сервис - Язык (Language)" появится новый язык: Spanish.

Тут я готов улышаться возмущенные возгласы: "Автор упрощает и замалчивает!!! А Help к программе как перевести?!?" Я готов дать содержательный ответ. Вот [здесь](#) лежит архив с двумя практически аутентичными текстами справки на английской и на эсперанто. Берите и переводите на желаемый язык. Для правки файла *.htm применяйте любой удобный для вас редактор. Я лично использую Microsoft Expression Web 4, и не надо за это кидать в меня камни, все в чем-то грешны... В том же архиве лежит маленькая утилита htm2chm для изготовления chm-файлов из веб-страниц. При изготовлении файла Справки выбирайте пункт меню "Обычный help". Имя файла справки должно быть Demagog-Langname.chm. Где Langname - это имя, выбранное вами для языкового файла *.In

Так, в нашем примере имя файла Справки будет Demagog-Spanish.chm.

Если же вы будете лениться и, переведя файл языковых ресурсов, не сделаете того же для Справки, то ничего страшного, вообще говоря, не произойдет. Если для некоторого языка интерфейса отсутствует Справка, то Demagog в таком случае покажет Справку на английском языке. И нечего возмущаться, сами виноваты. Так что, засучив рукава, принимайтесь за работу. Да, кстати. Не забудьте в переведенной вами Справке указать свое имя, дабы заслуженно прославить его 😊

<<

14. Путь самурая. Demagog в мире Юникода

Не претендующий на точность исторический экскурс. Давным-давно, когда компьютеры были большими 😊 мудрые люди придумали изображать каждую букву одним байтом. Байт - это 8 двоичных разрядов: 00000000 - пустой символ #0 ... 11111111 - символ #255 - буква "я" в русской кодировке ANSI. Всего возможных

комбинаций из 0 и 1 по восемь штук в ряду - 256. Вот и весь алфавит ANSI. От 0 до 127 - английские буквы, цифры и спецсимволы. От 128 до 255 - символы какого-нибудь национального алфавита. В русской ANSI есть все символы русского, украинского и белорусского алфавита. Но нет например, некоторых букв немецкого, испанского и т.д. Вот откуда термин: "кодировка страницы". Для каждого языка - своя.

Шло время. Еще более умные люди решили: а давайте обозначать каждый символ 2-мя байтами! Это уже 16 нулей и/или единиц, всего комбинаций 2 в 16-й степени = 65536. Уже пронумерованные первые 256 символов придется обозначать тоже 2-мя байтами, хотя достаточно одного. Один байт всегда будет нулевой. КОТОРЫЙ ИЗ ДВУХ? ПЕРВЫЙ ИЛИ ВТОРОЙ? Вот так кодировка Unicode сразу разделилась на две ветви: Unicode и Unicode Big Endian.

А время, по своей неизменной привычке, продолжало идти. Еще более гениальные люди предложили: а давайте первые 256 символов обозначать, как они уже обозначены: ОДНИМ байтом! А уже дальше - ДВУМЯ. Когда исчерпаются все 65536 2-байтовых комбинаций, то новые символы обозначать уже ТРЕМЯ байтами. 2 в 24-й степени комбинаций = 16777216. Если не хватит и этого, то для новых символов возьмем уже по 4 байта... Так появилась кодировка UTF-8. Как видно из вышесказанного: для английского языка она полностью совпадает с кодировкой ANSI.

В общем, стандарт Юникода получился достаточно сложным. И это я еще не всё рассказал... Тем не менее, такое направление оказалось наиболее жизнеспособным. Юникод был воспринят как стандартное средство передачи текста и данных. Им поддерживается фактически любая система письма в мире, поэтому Юникод является нормой в глобальном научно-техническом сообществе.

На сегодня Юникодом закодировано больше миллиона разных символов, охватывающих все языки мира. Самым важным преимуществом Юникода является то, что он позволяет представлять данные в текстовом формате без установки соответствия текстовых строк и информации о кодировке строки. Например: хранение и поиск текстов на русском, иврите, французском, испанском, немецком и английском языках. С Юникодом это сделать просто -

кодировка одна на всех. Потребуется добавить казахский или японский - без проблем!

Таковы мирового масштаба причины, побудившие меня создать новую версию Demagog, полностью поддерживающую Юникод 😊

Путь самурая. Если кому-то покажется, что это было совсем просто, то он немножко ошибается. Здесь я расскажу об этой увлекательной эпопее.

Не буду останавливаться на разных мелочах, которые в юникодной Delphi работают совсем не так, как ты привык. Сломался главный инстинкт программиста: один символ - это ровно один байт. Если в вашей программе есть модули, которые используют этот, переставший быть истинным факт - будьте любезны переделать. Сломался другой мощный инстинкт: конец строки в тексте обозначается парой символов #13#10 - возврат каретки + перевод строки. В Юникоде, кроме этой неразлучной пары есть и еще комбинации. В т.ч. и одиночный #13. Когда текст загружается в компонент RichEdit, то вместо каждой сладкой парочки остается один одинокешенек вышеупомянутый. Учтите это.

И, главное. Если что-то шустро работало на ANSI-алфавите в 256 символов, то где гарантия, что на необорзи... необозримом юникодном *алфавитище* это будет также. Некоторые умники предсказывали, что мой любимый хеш-алгоритм перестанет работать в Юникоде. То что летало - станет ползать. Я в это не верил, но некий внутренний трепет все же испытывал, признаюсь. Тестирование показало: если принять скорость хеш-алгоритма для кодировки ANSI за 1, то в Юникоде это время колеблется в пределах 1.105 .. 1.164. Нормальным языком: если и замедлилось, то не более чем на 17%. Ура, однако. А впрочем, я так и знал.

Если хеш-алгоритм не летал, а ползал в юникодной "Балаболке" (автору которой я [разрешил](#) одно время использовать его, наравне с методом прямого перебора), то теперь совесть моя спокойна.

И вот, преисполненный самых радужных надежд, окрыленный внезапным быстрым развитием блицкрига... э-э, миграции проекта Demagog на юникодную Delphi, я приступил к тестированию работы словарей формата REX, основанных на т.н. регулярных выражениях (PB). Мощнейшее средство для корректировки текстов, которое на заре создания проекта в 2007 году в Demagog напрочь отсутствовало. А в "Балаболке" было! Правда, с одной оговоркой: с русским языком PB не работают.

Помню тот день, 1 сентября 2010. Скачиваю в Инете знаменитый бесплатный модуль RegExpr.pas - давнее творение Андрея Сорокина, добавляю в нем к перечню буквенно-цифровых символов *буквы русского языка*, компилирую проект - вуа-ля, всё работает! Выкладываю версию, пишу на Форум, где разработчики TTS-софта и его пользователи вместе пасутся. Заодно предлагаю словарям регулярных выражений, вместо банального INI, дать расширение REX. От Regular **EX**pressions.

Давний участник Форума - автор "Балаболки" Илья Морозов сразу подает голос: "Взяли компонент Андрея Сорокина TRegExpr и дописали русские буквы в константу RegExprWordChars?" "Точно так", - отвечаю. "...Тоже думал над этим, но к окончательному выводу так и не пришел... Я готов перейти на использование TRegExpr - чтобы обеспечить единый способ обработки правил". Переименовать ini в gex он также согласился, чтобы в обеих программах было единообразие форматов словарей.

И вот, возвращаясь в своем рассказе в наши дни, к самому драматическому моменту. Преисполненный радужных и так далее надежд... запускаю тест словаря типа gex. Как-то он поведет себя в Юникоде? А поведение, мягко говоря, странное... "Фирменная" демагоговская зеленая полоска даже не ползет... а имитирует вековой рост сталактитов в пещерах. На глаз движение малозаметно. Короче: в Юникоде работа gex-словарей замедлилась в 10 .. 20 раз! Твою ж мать...

И винить некого. Сорокин в документации к RegExpr честно предупреждал, что для Юникода модуль не оптимизирован и работает крайне медленно. Кто хочет и может - усовершенствуйте. Все правильно - халявному коню в зубы не смотрят. Годы прошли, коня починить никто не захотел или не смог...

Тут меня осенило. В Delphi, начиная с версии XE есть свой собственный, "родной" модуль работы с PB. Читаю мануал, мне всё нравится, понятнее и проще, чем в модуле Сорокина. Uses RegularExpresions и... поехали! Что там, поехали... Понеслась птица-тройка! Потирая руки от радости, я наблюдал, как меньше чем за минуту был обработан текст в 2.8 Мб. Лишь один маленький нюанс, маленькая бочка дегтя в ложке меда отравила мое ликование. Результат оказался неверным! Потому что метасимволы \w \W \b \B - не реагировали на русские буквы! Старинная традиция в системах обработки PB: "буквами" считаются только буквы латинского алфавита и цифры. Англосаксы изобрели теорию регулярных выражений, они же и установили правила. Остальные аборигены земного шара нервно курят в сторонке.

Поиск решения проблемы в Инете выдал только скулеж и плач на разных форумах: как быть и что делать с кириллицей в регулярных выражениях?! Ахи да охи на разных языках... Сопли и слезы в три ручья. Твою ж мать...

Странствия в Инет-пространстве длились уже третий день. Тотальный обыск земного шара вразумительных результатов не давал. Поздно вечером, возлежа, подобно турецкому паше, на диване, я уныло глядел в экран планшета. Может, ну его, лучше фильм посмотреть? Если бы мой верный планшет не спросил: "Язык сайта японский, перевести на русский?" и если бы я не ответил: "Да", то я бы не понял, что нашел.

Библиотека поддержки PB для Delphi 2005 и выше: SkRegExpr. Автор: Shuichi Komiya. Поддержка Юникода. Совместимость со стандартом PCRE 5.14. Совместимость с "родным" модулем поддержки PB в Delphi XE. Совместимость на уровне функций и методов с модулем RegExpr Андрея Сорокина. Но это - самостоятельная разработка, а не продолжение сорокинской, оговаривался Shuichi Komiya. И рекомендовал всем, кто хочет идти по его стопам, книгу Yoshiyuki Kondo "Algorithms and Data Structures for C Programmers". Из которой он заимствовал алгоритм построения недетерминированного конечного автомата для регулярных выражений. (Не пугайтесь, это всего лишь математика). Недостаток: документация только на японском. Не беда. Гуглопереводчик нам в помощь.

Распространение: бесплатно. Программный продукт защищен лицензией MPL ([Mozilla Public License](#)). Это означает: **исходный код, скопированный или измененный под лицензией MPL, должен лицензироваться по правилам MPL.** Бери, пользуйся, переделывай, если надо, включай в любой свой проект, хоть коммерческий, хоть нет. При условии, что объявишь в своем проекте факт использования MPL-защищенного программного обеспечения и выложишь в открытый доступ его исходные коды под той же лицензией. Как говорится: попользовался сам, передай другим. На своем сайте Shuichi Komiya указывал, что заинтересован в широком распространении SkRegExpr.

"Моя надежда" - программа, написанная на базе SkRegExpr, переводит на русский язык...

ура, наверное... - подумал я, написал `uses skregularexpressions...`, проверил на маленьком примере - с русским языком всё работает нормально, а как, например с эсперанто?! Тоже алфавитно-цифровые символы и границы слов отлавливаются нормально, аборигены земного шара аплодируют стоя. "В самом деле, ура!" - решил я и запустил сакраментальный тест - 2.8 Мб. Успел заметить, что скорость практически такая же, как в "родном" дельфийском модуле PB. А потом, примерно на 40% готовности, чудесный "самурайский модуль" завис намертво. Твою ж мать...

Место, где происходит катастрофа я нашел быстро, и выявил своеобразный класс регулярных выражений, убивающих творение Shuichi Komiya наповал. Написал короткое консольное приложение, демонстрирующее правильный ответ для "родного" модуля PB и бесконечный цикл для библиотеки SkRegEx. И отправил на сайт разработчика в раздел "Извещения об ошибках".

Новую, исправленную версию библиотеки SkRegEx, и объяснение, в чем состояла ошибка, Shuichi Komiya выложил через 2 часа 23 минуты, продемонстрировав оперативность и высокий профессионализм. Остальные проверки на текстах большого объема выполнил **Евгений Мирошниченко**, всё отработало нормально и с высокой скоростью.

Исходные коды SkRegEx, слегка измененные мною (чтобы стандартные сообщения выдавались не на японском, а на английском языке) содержатся в дистрибутиве Demagog. Они защищены лицензией [MPL 2.0](#), см. выше. Кому надо, пользуйтесь, с соблюдением условий лицензии.

Там чудеса, там леший бродит... или ловушка #128. Как я уже говорил, Delphi-программиста, работающего с юникодной версией Delphi, могут подстергивать неожиданные засады. Здесь я расскажу еще об одной.

Однажды Шуичи Комия написал в своем блоге следующее.

"Я думал, что `#0080` и `#128` - это одно и тот же. Но столкнулся с ситуацией, когда это, кажется, не так. Результат сравнения в SkRegEx отличался у зарубежного пользователя. Регулярное выражение: `\b[Ĉ]([Ĉd]+)`, строка для поиска: `Ĉ ĈĈĈ`. Правильный ответ: соответствие двух первых символов. Это очевидно и так и получается в моем тесте. Но в тесте пользователя соответствие не было найдено. Хуже всего было то, что ошибка не

воспроизводилась и это серьезно беспокоило меня. К счастью, Пользователь-сан согласился сотрудничать, чтобы решить проблему. Много раз, чтобы определить проблемные участки кода, приходилось отправлять тестовую программу и получать результат от пользователя. Код, вызвавший ошибку, оказался таким:

```
if AStr^ < #128 then
```

Для `Ĉ` с кодом U+0108H это должно быть False. Тем не менее, результатом тестового кода стало True. Проблема была решена изменением кода на

```
if AStr^ < #$0080 then
```

Но, в чем же было дело?" Такой резонный вопрос задавал Шуичи Комия. А собака, как выяснилось, была зарыта вот где. По крайней мере в Delphi XE8 десятичная и 16-ричная запись кода символа могут означать *разные символы!* [Разъяснение от разработчиков Delphi](#) Круто, да? Десятичная запись `#xxx` - это символ ANSI. 16-ричная запись из двух цифр `#$xx` - это символ ANSI. 16-ричная запись из четырех цифр `#$xxxx` - это WideChar. В результате, запись `#128` дает разные символы в японской и русской кодовых страницах ANSI. Что и было обнаружено при тестировании новой версии SkRegEx.

Как вы, наверно, догадались, "Пользователь-сан", составивший контрольный пример - это был я 😊

<<

15. "Брюки превращаются..." или что такое "фонетический алфавит"?

Как всё начиналось. В 1886 г. группа британских и французских преподавателей задалась целью разработать алфавит для передачи устной речи на любом языке. Их труд увенчался успехом, а рабочая группа превратилась со временем в Международную фонетическую ассоциацию. Более ста лет она занимается поддержкой и развитием Международного фонетического алфавита. Сокращенно - МФА или, в английском написании - IPA. От International Phonetic Alphabet. Кто изучал английский язык, наверняка помнит замысловатые "знаки транскрипции". Это МФА и есть.

На основе МФА созданы алфавиты для некоторых, ранее бесписменных языков. Профессиональные оперные певцы, исполняющие арии на многих языках, широко пользуются МФА. Естественно, что МФА полезен при синтезе речи, когда требуется исправить произношение того или иного слова, сообщив голосовому движку его правильное произношение. Но как это сделать? Как составить правило корректировки произношения с помощью МФА, большинство знаков которого отсутствует на клавиатуре! Будьте любезны, убедитесь воочию.

br^j ' ʌ k^j I pr^j Iv re^ç : ' æ j ʊ t s ə
v[˘] t^j I g^j ' a n t n t j ə s^j ' o r t^j

Родство с латиницей явно прослеживается. Но, чтобы передать особенности произношения любого из языков мира, пришлось к 26 буквам добавить еще в три раза больше. Каждый новый символ всё менее и менее походил на прежние. В ход также пошли надстрочные и подстрочные значки, т.н. диакритики. Сложное дело - всемирная азбука! Приглядитесь: в ней даже "двоеточие" и "апостроф" совсем не такие, как на клавиатуре. Но задачу: научить компьютер понимать фонетическую запись - никто не отменял.

Решение оказалось банальным. Если использовать латинские буквы, цифры и прочие специальные клавиатурные символы не только по одному, а в и комбинациях по два, то вот вам, пожалуйста. Каждому символу МФА найдется соответствие. На глаз запись выглядит тоже непрезентабельно, ну, да компьютеру всё равно. Так появился машиночитаемый фонетический алфавит Speech Assessment Methods Phonetic Alphabet (SAMPA) и его расширение X-SAMPA. Некоторые голосовые движки, имеющие собственную систему корректировки произношения, как раз используют SAMPA.

используют SAMPA для этой цели. К ним, например, относятся русскоязычные голоса: ASARELA Anyona, IVONA Tatyana, IVONA Maxim.

Вот и всё про SAMPA, потому что тег SAPI5 <PRON SYM="..."/>, предназначенный как раз для чтения фонетической записи, этого алфавита не понимает. Так какого рожна ему надо?! Тут-то и начинается самое интересное. Тег <PRON> понимает лишь то, что для него придумали затейники из Майкрософт. А придумали они по-первоначально [SAPI Phone Set](#) - набор фонетических алфавитов для нескольких, самых распространенных языков. Английский, китайский, японский... полный список (он невелик) вы найдете на сайте Майкрософт. Каждая фонема того или иного языка изображается некоторой комбинацией маленьких латинских букв.

Долго ли, коротко ли, но этим дело не ограничилось. Последовала новая разработка: [Microsoft Universal Phone Set](#) (UPS), полностью основанная на МФА. Фонемы изображаются большими латинскими буквами или их комбинациями

по две и по три. Диакритики выглядят как комбинации из трех маленьких латинских букв. Используются также цифры 1, 2; знак подчеркивания _; вертикальная черта |; и знак +. UPS применимо для всех голосов, кроме тех, которые заточены под SAPI Phone Set, см. выше. Фонемы на письме разделяются пробелом.

На первый взгляд UPS-транскрипция вызывает оторопь. Да, согласен: и на второй тоже. Но, опять же, компьютеру всё равно!

Входящие в состав Windows 10 искусственные голоса: MS Irina Desktop - Russian и MS Pavel Mobile - Russian - поддерживают тег <PRON> с фонетикой UPS. Больше того, это верно и для упомянутых выше IVONA Tatyana, IVONA Maxim!

Что открывает возможность управлять их произношением исключительно средствами SAPI5, вставляя в читаемый текст тег <PRON>. Для этой цели в Demagog есть опция "Правка - Фонетическая транскрипция (Shift+F1)". Фонетическая таблица-подсказка поможет составить фонетическую запись того или иного слова или даже целой фразы. Можно просто вставлять из буфера обмена транскрипцию МФА (где вы ее возьмете, это другой вопрос) и легким движением руки перевести в UPS.

Но проще будет воспользоваться имеющимся в Demagog автоматическим переводом русских слов в транскрипцию МФА. Он не обеспечивает абсолютной точности, но дает вполне удовлетворительный результат. Который, если захотите, сможете подправить вручную. Русское слово/словосочетание следует вводить в *нижнем* регистре, а ударные гласные обозначать *верхним* регистром!

| UPS | IPA | Пример звучания или комментарий |
|-----|-------|---------------------------------|
| P | [p] | put |
| B | [b] | big |
| M | [m] | mat |
| BB | [v] | |
| PH | [ʃ] | |
| BH | [β] | cabra |

Нажав на кнопку "RU -> IPA", получим транскрипцию МФА.

| UPS | IPA | Пример звучания или комментарий |
|-----|-------|---------------------------------|
| P | [p] | put |
| B | [b] | big |
| M | [m] | mat |
| BB | [v] | |
| PH | [ʃ] | |
| BH | [β] | cabra |

Вторая кнопка превратит ее в транскрипцию UPS. А кнопочка внизу слева запускает проверку: читается ли фонетическая запись так, как мы хотим?

Язык голосового движка: Russian

☒ UPS <PRON SYM="%1"/>
 ☐ PLS <phoneme alphabet="ipa" ph="%2"/>

Фонемы, которые будут помещены в документ

B RR pal S1 YX K pal IH P RR pal IH V RR AEX SC lng S1 AE J YX TS AX V + IX low L pal IH G S1 A N T N IX I

| UPS | IPA | Пример звучания или комментарий |
|-----|-------|---------------------------------|
| P | [p] | put |
| B | [b] | big |
| M | [m] | mat |
| BB | [ʋ] | |
| PH | [ɸ] | |
| BH | [β] | cabra |

Если да, то жмем кнопку с галочкой и в текст с позиции курсора вставляется тег SAPI5: <PRON SYM="B RR pal S1 YX K pal IH P RR pal IH V RR AEX SC lng S1 AE J YX TS AX V + IX low L pal IH G S1 A N T N IX low J AX SR S1 O RR T IX low"/> [Вот как это звучит](#)

"Автор упорно не договаривает!!" - слышу чей-то возмущенный голос. - "Что это за странная аббревиатура PLS на скриншоте, про которую, однако, не сказано ни слова?!" Что ж. Как выражался знаменитый профессор Выбегалло, "даю пояснения..."

[Pronunciation Lexicon Specification](#) (PLS) - это стандарт XML-разметки текста, который предлагается использовать для компьютерных программ распознавания и синтеза речи. Он рекомендован [Консорциумом Всемирной Паутины](#) (W3C). Встроенные системы коррективки произношения многих современных голосовых движков, (например, IVONA Tatyana, IVONA Maxim), как раз основаны на PLS.

В этом стандарте, для указания произношения того или иного слова, используется Международный Фонетический Алфавит. Так сказать: весомо, грубо, зримо - никаких заумных вывертов с "машиночитаемой записью". Современные компьютеры понимают Юникод, и следовательно, любой алфавит. В том числе и МФА! Хотя, наряду с ним, в PLS допускается использование и символов алфавита X-SAMPA.

Вот этот тег PLS движок Maxim (или Tatyana) прочтет без всякого труда. Звучание будет точно такое же четкое и правильное, как по ссылке выше.

<phoneme alphabet="ipa" ph="bri'ʊki priivres'æjʊtsə v_ɪlɪg'antnɪjə ʃ'ortɪ"/>

Майкрософт отдыхает, нервно покуривая в сторонке... (шучу 😊).

Получение транскрипции с помощью встроенного интерпретатора. Пример использования функций RuIPA и RuUPS. Нижеприведенный скрипт Phonetic Dictionary RU.lua должен находиться в папке _Tests_. В активной вкладке должен находиться список русских слов - все буквы в нижнем регистре, а ударные гласные - в ВЕРХНЕМ регистре. Вызов скрипта через меню по Ctrl+F2. Во вкладке "0 - Статистика" будет сформирован соответствующий dic-словарь - его можно сохранить под каким-либо именем и с расширением .dic.

```
-- Создание фонетического dic-словаря в окне "0 - Статистика"
-- по списку слов с ударениями, находящемуся в активной вкладке.
-- Например:
-- выкатившиеся
-- двадцати
-- темно
```

```
os.setlocale("", 'ctype') -- национальная кодовая страница
```

```
cap = 'Укажите тип транскрипции'
items = {
    'Международный фонетический алфавит (IPA)',
    'Microsoft Universal Phone Set (UPS)'
}
```

```
ans = Menu(cap,items)
if ans == 0 then goto HALT end
if ans == 1 then
    -- шаблон тега Pronunciation Lexicon Specification
    mask = '<phoneme ph="@"/>'

    else
        -- шаблон тега Microsoft SAPI5
        mask = '<PRON SYM="@"/>'
    end
```

```
ind = WActive() -- номер активной вкладки
bom = '\239\187\191' -- спецификация UTF-8
s = WText(ind)
a = string.split(s,'\r')
c = ""
for i = 1,#a do
    if a[i] ~= "" then
        if ans == 1 then d = RuIPA(a[i]) else d = RuUPS(a[i]) end
        b = string.lower(a[i])
        b = AnsiToUtf8(b)
        c = c..b..' '..string.gsub(mask,'@',d)..' \r'
    end
end
-- передадим новый utf8-текст во вкладку 0
-- через сохранение во временном файле
c = bom..c
```

```
SaveToFile({c}, '_tmp')
WOpen(-1, '_tmp'); WNew(0); WAdd(0, -1, '\r')
os.remove('_tmp')

WActive(0)

::HALT::
os.setlocale('C')
```

<<

16. Формат DXT - "документ Demagog"

Изначально Demagog работал только с т.н. "плоскими" текстами. Как Блокнот Windows, проще говоря. А прочие поддерживаемые форматы на лету преобразовывал в этот самый плоский до банальности текст. Ни тебе курсива, ни жирного шрифта. Выделить что-то подчеркиванием... тоже низзя! аяяй!

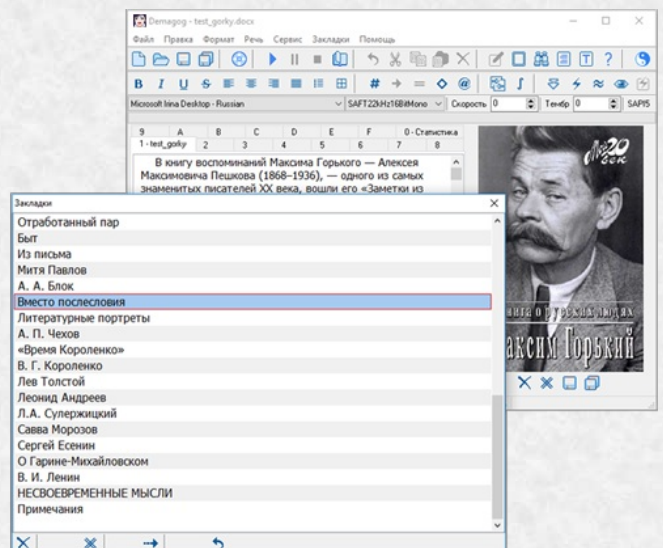
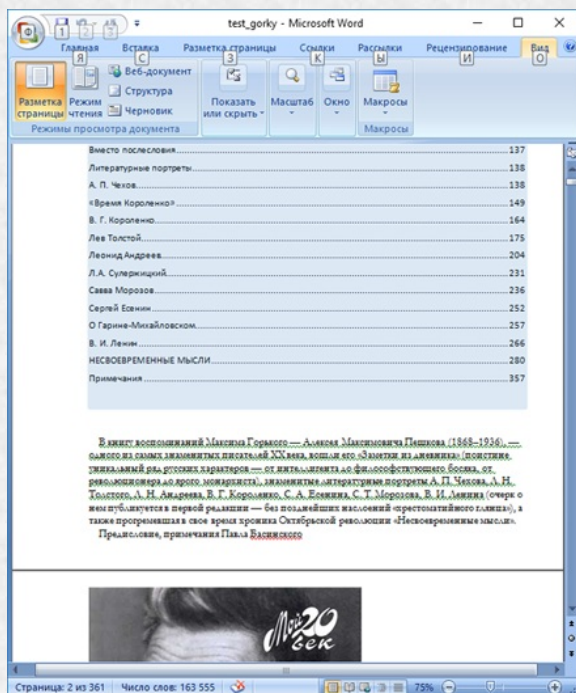
Созданный пользователем перечень закладок хранился в файле настроек Demagog - \$.cfg. При его порче или случайном удалении пропадали результаты долгого, кропотливого труда по разметке текста закладками. А картинки, импортированные из документов MS Word или электронных книг, хранились в автоматически создаваемых папках-галереях. Которые Demagog, как корова лепешки, оставлял везде, где пасся... то бишь, где открывал какие-либо файлы.

"Работа проделана большая, но дальше так дело не пойдет", - подумал я и задействовал поддержку программой собственного текстового формата. Demagog teXT - DXT. (Название и сама идея добавления этого формата в программу были ранее предложены **Евгением Мирошниченко**). Плоский текст по-прежнему поддерживается, но картинок за собой в котомке больше не носит. Закладки, как и раньше, хранятся в настойках Demagog, но...

Файл формата DXT хранит в себе, помимо текста: упомянутый список закладок; а также импортированные картинки. При редактировании текста поддерживаются общепринятые стили: выравнивание влево, вправо, по центру. Шрифт: обычный, жирный, курсив, подчеркнутый, зачеркнутый. Допускается совмещение в одном тексте фрагментов, написанных разными шрифтами.

Любая часть текста может быть оформлена в виде т.н. "маркированного списка". Доступно создание простых таблиц, типа "шахматки".

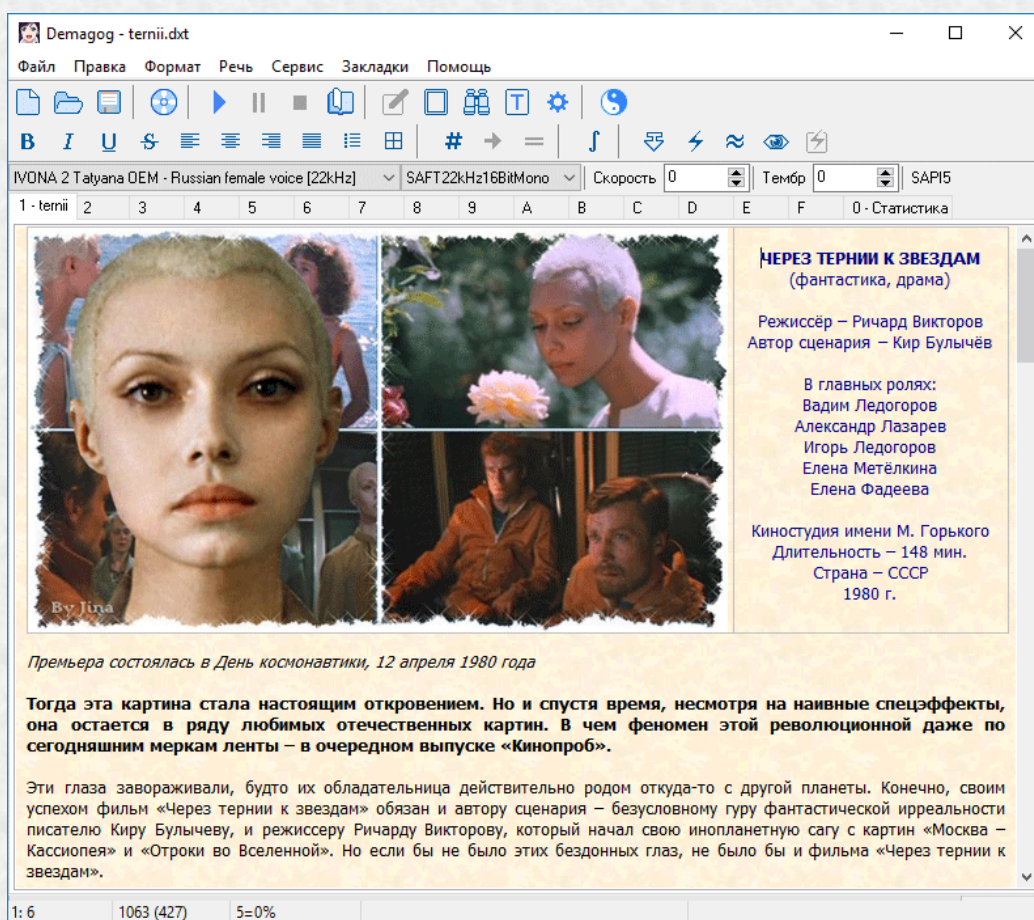
По предложению **Евгения Мирошниченко** (подготовившего также соответствующие контрольные тесты) реализовано автоматическое распознавание "внедренного оглавления" в документах MS Word - .docx. Если такое оглавление присутствует в docx-файле, то оно будет преобразовано в список закладок Demagog.



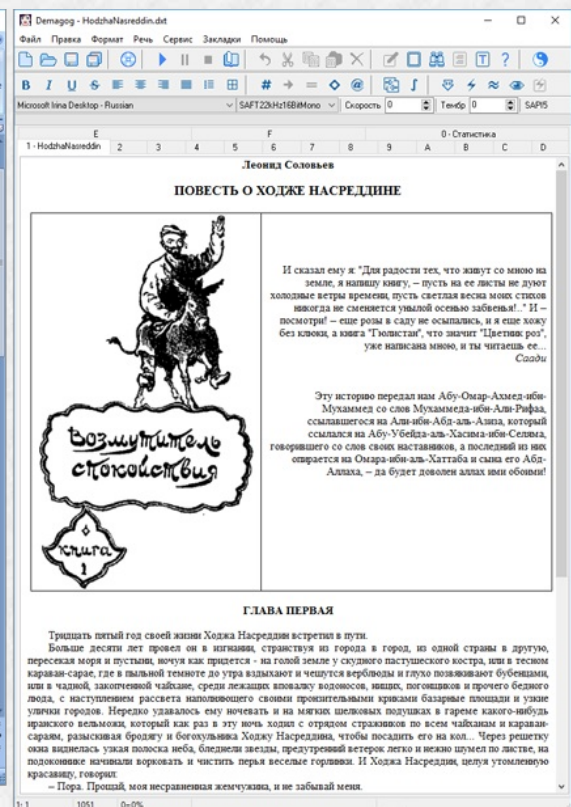
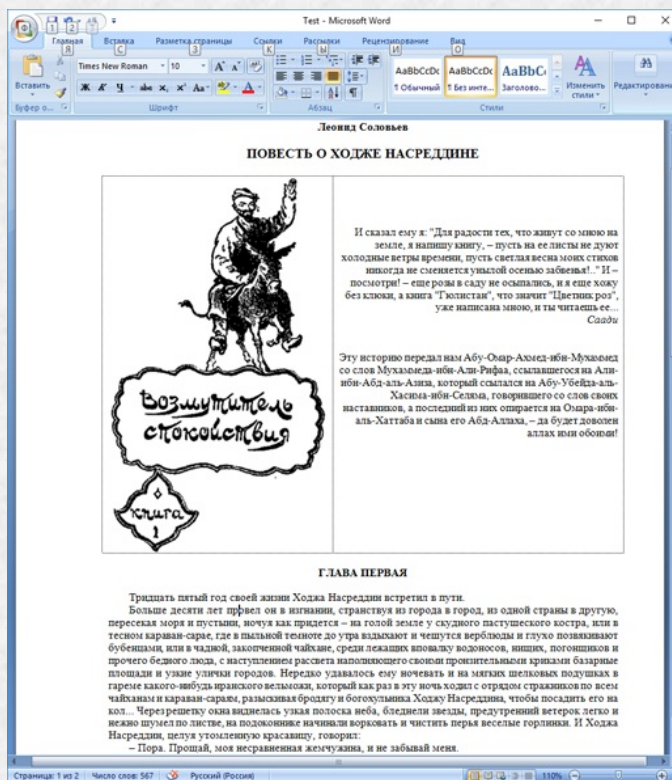
Если текст имеет закладки и/или импортированные картинки, то при его сохранении в Demagog, по умолчанию предлагается формат DXT.

DXT, в основном, соответствует спецификации RichEdit 4.1 - подмножеству известного межплатформенного формата RTF. Demagog дает возможность сохранять DXT -> RTF. За исключением Галереи импортированных картинок и списка закладок Demagog - эти данные игнорируются. В этой части форматы DXT и RTF несовместимы.

Под Windows 10 для формата DXT дополнительно доступны: выравнивание текста по ширине; вставка картинок прямо в текст перетаскиванием или из буфера обмена; изменение ширины столбцов таблицы перетаскиванием их границ мышью.



Документы MS Word, перенесенные из буфера обмена (например, в контекстном меню "Вставить, как новый документ", имеют вполне приемлемый вид.



Как видим, основные элементы форматирования совпадают. Такой dxt-файл сохраняется в rtf вместе с имеющимися в нем картинками.

<<

17. Извлечь шляпу из кролика... или текст любой ценой

Что делать, если надо прочесть текст, формат которого непонятен Demagog?

Вариант 1. Написать гневное письмо автору, мол немедленно включите в свою программу поддержку файлов формата ".hz", ".bla", ".hren" и т.д. На что автор ответит, что это противоречит принципу минимализма, и, вообще, он - не раб на галерах.

Вариант 2. Найти в Инете среди великого множества свободно-распространяемого ПО консольный конвертер, который принимает на входе ваш файл, и создает новый файл, содержащий только текст. Что-то вроде: **AnyConverter.exe "Важный документ.hren" "Важный документ.txt"**.

Что? Не нашли ничего подобного? А значит, этот формат никому не известен и никому не нужен. А вот погуглите "pdf to txt" и сразу поймете всю важность проблемы с форматом PDF. Его даже MS Word не откроет, нужно установить Adobe Acrobat - такого же звероподобного монстра. Цирк, да и только. Так чего же вы хотите от Demagog? Аа-а... текст любой ценой! Вынь, да положь.

Начиная с версии 7.28.320 Demagog поддерживает подключение внешних консольных текстовых конвертеров.

При записи настроек в файле **..\profiles\aliens.lst** применяются следующие обозначения:

%1 - полное имя исходного файла

%2 - полное имя выходного текстового файла, такое же, как %1, но с расширением .txt

%~p1 - путь к файлу из %1

%~n1 - короткое имя файла из %1 (т.е. без пути и без расширения)

Упомянутый "условный конвертер" выглядел бы в настройках так:

Хреновые файлы (.hren ...)|.hren|.hre|.hru=AnyConverter.exe %1 %2

Сперва идет поясняющий текст - та строчка, которая будет показана в диалоге открытия файлов. После вертикальной черты - перечень расширений, которые относятся к указанному типу файлов. Разделитель - опять же вертикальная черта. И, в завершение, после знака равенства - собственно командная строка для конвертера. Указание ему, что надо делать, чтобы создать временный текстовый файл в кодировке utf-8, с тем же именем и в том же месте, где исходный документ.

Конвертация документа в плоский текст происходит незаметно для пользователя, и текст в окне Demagog появляется так же быстро и непринужденно, как если бы его извлек из документа сам Demagog.

В зависимости от конвертера, запись командной строки может выглядеть изящно и коротко, или быть более громоздкой. К примеру, xdoc2txt.exe, входящий в дистрибутив Demagog, *автоматически создает текстовый файл с тем же именем*. И командная строка выглядит довольно просто: `Adobe PDF (.pdf)|.pdf=xdoc2txt.exe -f -8 %1`

Конвертер совсем не обязательно помещать в рабочую папку Demagog. Можно просто указать в настройках полный путь к его исполняемому файлу. Вот пример использования portable-программы [Text Mining Tool](#), в состав которой входит консольная утилита преобразования файлов .pdf, .doc, .rtf в плоский текст:

Adobe PDF (.pdf)|.pdf=d:\2 - Install\Misc programs\Text Mining Tool 1.1.42\minetext.exe %1 %2

Серым цветом показан путь к консольной утилите, как он выглядит на моем компьютере. Аналогичный пример для утилиты [blb2txt](#):

Adobe PDF (.pdf)|.pdf=d:\000\blb2txt.exe -f %1 -v %~p1 -p %~n1 -e "utf8"

А вот пример для утилиты [pdftotext](#), если ее поместить в корневую папку Demagog:

Adobe PDF (.pdf)|.pdf=pdftotext.exe -q -nopgbrk -enc UTF-8 %1 %2

И т.д. и т.п. Принцип, полагаю, понятен.

В настройках можно прописать любое количество конвертеров, каждый из которых будет обрабатывать определенные типы файлов. Если конвертер не найден, то соответствующие ему строки в `aliens.lst` - игнорируются. Если нет ни одного конвертера из перечисленных в настройках, то `aliens.lst` игнорируется целиком.

<<

18. Великан на дороге, или нейросети для синтеза речи

Время идет... и компьютерные технологии развиваются. На сегодняшний день получила популярность такая услуга, как *синтез речи онлайн*. Некоторые из этих сервисов уже используют новейшую технологию [нейросетей](#) для синтеза речи, практически неотличимой от человеческой. Для этого, конечно, требуются значительные вычислительные ресурсы, которыми заведомо не обладает домашний компьютер.

Здесь я приведу небольшой список онлайн-сервисов. Тех, которые заметны на русскоязычном пространстве, и известны мне. Все они - платные, но позволяют бесплатную озвучку в ограниченном объеме в демонстрационных целях.

| | |
|--------------------------------|---|
| Google Cloud Text-to-Speech | https://cloud.google.com/text-to-speech/ |
| Yandex Speech Kit | https://cloud.yandex.ru/services/speechkit |
| Amazon Polly | https://aws.amazon.com/ru/polly/ |
| Центр Речевых Технологий (ЦРТ) | https://cloud.speechpro.com/service/tts |
| VoxWorker | https://voxworker.com/ru |
| [vs]robotics | https://vsrobotics.ru/products/speech-synthesis/ |
| и др. | ищите в Инете по запросу "синтез речи онлайн" |

Скорее всего, с удешевлением услуг онлайн-озвучки, облачные технологии полностью вытеснят десктопные варианты Text-To-Speech. Или же... дальнейшее развитие вычислительной техники позволит энтузиастам создавать "нейросети на дому". И тогда нас ждет раздолье и обилие самых разнообразных "искусственных", но при том естественно-звучащих голосов. Будущее покажет. Впрочем, оно уже наступает...

Silero. Чем занимается эта команда - станет понятно, если пройти по ссылке. В том числе они опубликовали в свободном доступе качественную систему синтеза речи. Она основана на специально обученной нейросети. Структура сети не раскрывается, но основанная на ней модель, т.н. V3.1 доступна для скачивания и использования локально. Основные характеристики:

- естественная речь;
- радикальная простота и минимализм;
- высокая скорость работы на 1 потоке / ядре процессора;
- не требует наличия видеокарты на компьютере;
- поддержка синтеза в разном качестве;
- наличие уникальных голосов;

Статьи разработчиков, опубликованные на [Хабре](#), вызвали волну восторженных откликов, среди которых, впрочем, слышались недовольные голоса. "Ни хрена не понятно!" "Ни фига не работает!" "Python - гоно!" и т.п.

Ниже я приведу [рабочий пример](#) питон-скрипта от Silero. Для его запуска на компьютере должен быть установлен Python 3 с установленным в нем PyTorch 1.10+. Кстати, Python 3.10 у меня уже был установлен, а торч - нет.

Не беда. Идем на [оф. сайт](#) и выбираем в меню нужные нам пункты, см. картинку:

| | | | | | | |
|-------------------|---|-----------|-----------|------------|------------|--------|
| Your OS | Linux | | Mac | | Windows | |
| Package | Conda | | Pip | | LibTorch | Source |
| Language | Python | | | | C++ / Java | |
| Compute Platform | CUDA 10.2 | CUDA 11.3 | CUDA 11.6 | ROCm 5.1.1 | | CPU |
| Run this Command: | pip3 install torch torchvision torchaudio | | | | | |

Внизу меню показывается, какую утилиту Python надо выполнить, чтобы установить из Интернета PyTorch с выбранными опциями. Запуск производится из консоли CMD! Например, таким bat-файлом:

```
c:\Python310\Scripts\pip3 install torch torchvision torchaudio
pause
```

Серым шрифтом показан путь к утилите pip3.exe на моем компьютере. У вас он будет другим. Пакет PyTorch имеет (в распакованном виде) размер около 1 Гб, и его установка займет некоторое время. Процесс установки отображается на консоли. Когда установка завершится, тестовый скрипт уже можно будет запускать.

```
# V3
import os
import torch

device = torch.device('cpu')
torch.set_num_threads(4)
local_file = 'model.pt'

if not os.path.isfile(local_file):
    torch.hub.download_url_to_file('https://models.silero.ai/models/tts/ru/v3_1_ru.pt', local_file)

model = torch.package.PackageImporter(local_file).load_pickle("tts_models", "model")
model.to(device)

example_text = 'В недрах тундры выдры в г+етрах т+ырят в вёдра ядра к+едров.'
sample_rate = 48000 #8000 24000 48000
speaker='baya' #'aidar' 'baya' 'kseniya' 'xsenia' 'eugene' 'random'

audio_paths = model.save_wav(text=example_text, speaker=speaker, sample_rate=sample_rate)
```

Запуск аналогично, по-рабоче-крестьянски, bat-файлом:

```
c:\Python310\python.exe c:\Users\alloys\Downloads\silero1.py
pause
```

Серым шрифтом показаны пути на моем компьютере, у вас они будут другие. После запуска на экране появляется консоль CMD с возрастающими циферками процентов - идет скачивание модели. На 100% возникает пауза в несколько секунд, и Press any key...

В папке запуска обнаруживается файл test.wav с озвучкой тестовой фразы, и файл model.pt - модель V3.1. Эта модель содержит 5 русскоязычных дикторов, выбирайте любой. Теперь оператор if можно из питон-скрипта убрать - ведь мы уже получили экземпляр модели на свой компьютер.

```
# V3
import os
import torch

device = torch.device('cpu')
torch.set_num_threads(4)
local_file = 'model.pt'

model = torch.package.PackageImporter(local_file).load_pickle("tts_models", "model")
model.to(device)

example_text = '<s><s>Наши цели ясны, задачи определены.</s><s>За работу, товарищи!</s><s>За новые победы коммунизма!</s></s>'
sample_rate = 48000 #8000 24000 48000
speaker='eugene' #'aidar' 'baya' 'kseniya' 'xsenia' 'eugene' 'random'

audio_paths = model.save_wav(ssml_text=example_text, speaker=speaker, sample_rate=sample_rate)
```

В новой тестовой фразе использованы теги SSML. Как уверяют разработчики, модель их поддерживает. Посмотрим-посмотрим... Да, так и есть! Не забывайте только вместо параметра text указывать ssml_text. Более [выразительный пример](#):

```
# V3
import os
import torch

device = torch.device('cpu')
torch.set_num_threads(4)
local_file = 'model.pt'
```



```

model = torch.package.PackageImporter(local_file).load_pickle("tts_models", "model")
model.to(device)

ssml_sample = """
< speak>
< p>
    Когда я просыпаюсь, <prosody rate="x-slow">я говорю довольно медленно</prosody>.
    Потом я начинаю говорить своим обычным голосом,
    <prosody pitch="x-high">а могу говорить тоном выше </prosody>,
    или <prosody pitch="x-low">наоборот, ниже</prosody>.
    Потом, если повезет - <prosody rate="fast">я могу говорить и довольно быстро.</prosody>
    А еще я умею делать паузы любой длины, например две секунды <break time="2000ms"/>.
< /p>
    Также я умею делать паузы между параграфами.
< /p>
< p>
    <s>И также я умею делать паузы между предложениями</s>
    <s>Вот например как сейчас</s>
< /p>
< /p>
< /speak>
"""

```

```

sample_rate = 48000 #8000 24000 48000
speaker='baya' #'aidar' 'baya' 'kseniya' 'xenia' 'eugene' 'random'

```

```

audio_paths = model.save_wav(ssml_text=ssml_sample, speaker=speaker, sample_rate=sample_rate)

```

Теперь те, кто знает питон, могут переделывать эти примеры, как им угодно, но в соответствии с лицензией [CC BY-NC-SA](#). Это означает возможность воспроизводить и делиться Лицензируемым материалом, полностью или частично, только для некоммерческих целей. А так же производить, воспроизводить и обмениваться адаптированными материалами только для некоммерческих целей. А в остальном, дерзай, выдумывай, пробуй...

В общем, если в новых победах коммунизма имеются определенные сомнения, то остальные слова товарища Хрущева, [по-прежнему, актуальны](#).

Demagog-x64-Silero. Воспользовавшись приведенными выше примерами, пользователь tonio_k создал сборку [Demagog для models silero от tonio_k](#) для обычной, 32-разрядной версии Demagog. Средствами встроенного интерпретатора Lua создается питон-скрипт, который передается на исполнение интерпретатору python. Питон, при этом, вызывается, как внешняя программа. В состав сборки входит переносимая версия питон, поэтому эта сборка полностью независима от наличия или отсутствия на компьютере пользователя установленного интерпретатора питон.

Можно ли, что-то подобное сделать для 64-разрядной версии Demagog? Очевидно, можно. Однако, можно сделать и по другому. Ведь Demagog-x64 умеет исполнять питон-скрипты "напрямую"! Итак, перед нами две проблемы.

1. Крайне желательно, чтобы использовался встроенный в эту версию Demagog embedded python. Но... embedded python не допускает инсталляцию сторонних пакетов! Как же установить в него пакет torch?!

2. Для удобства пользователей, питон-скрипт должен иметь графический интерфейс. Но, в embedded python отсутствует стандартная графическая библиотека tkinter. А установить ее, или какую-то другую - невозможно по причине... см. п.1.

Решение 1. С помощью установленного на компе полнофункционального питона, скачиваем пакет pytorch во временную папку, например bat-файлом:

```

C:\Python310\Scripts\pip3 install -t D:\1 torch torchvision torchaudio
pause

```

Серым цветом показаны установленный питон и временная папка на моем компьютере. У вас они будут другие. После того, как torch будет целиком скачан во временную папку, делаем следующее. В папке со встроенным питоном ..\Demagog\python создаем папку Lib. И всё содержимое временной папки переносим в папку Lib. Затем редактируем файл python310._pth, добавляя в его текст одну короткую строчку :

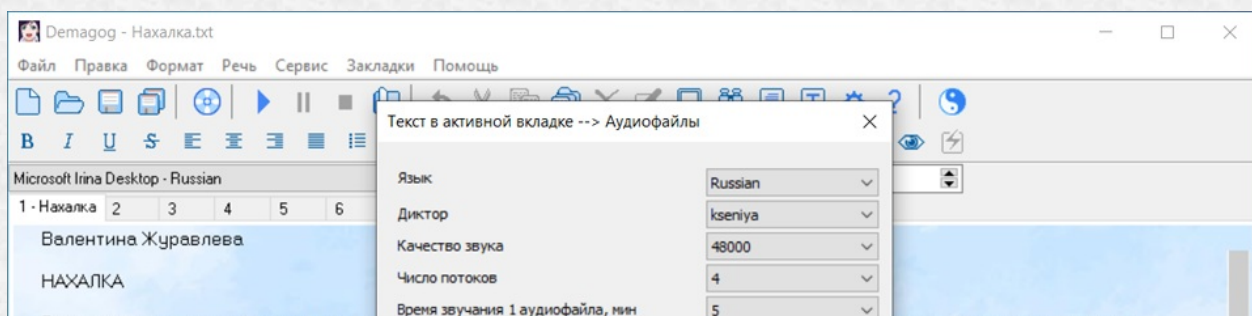
```

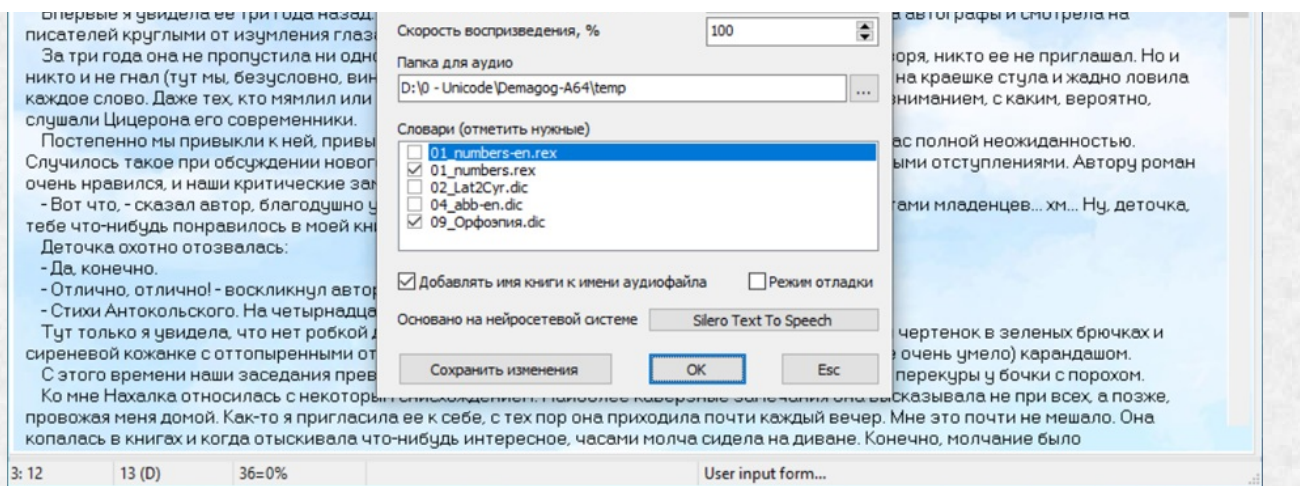
python310.zip
.
Lib
# Uncomment to run site.main() automatically
#import site

```

На этом - всё. Мы впихнули невпихуемое, и теперь наш встроенный питон содержит в себе пакет torch. Этот фокус работает для Windows 10 64x и Python 3.10.

Решение 2. Я сделал доступными для питон (хоть встроенного, хоть внешнего), минимально-необходимый набор функций Demagog. Все необходимые графические элементы интерфейса там есть. Оставалось написать сам питон-скрипт, что и было сделано. Вот, как это выглядит:

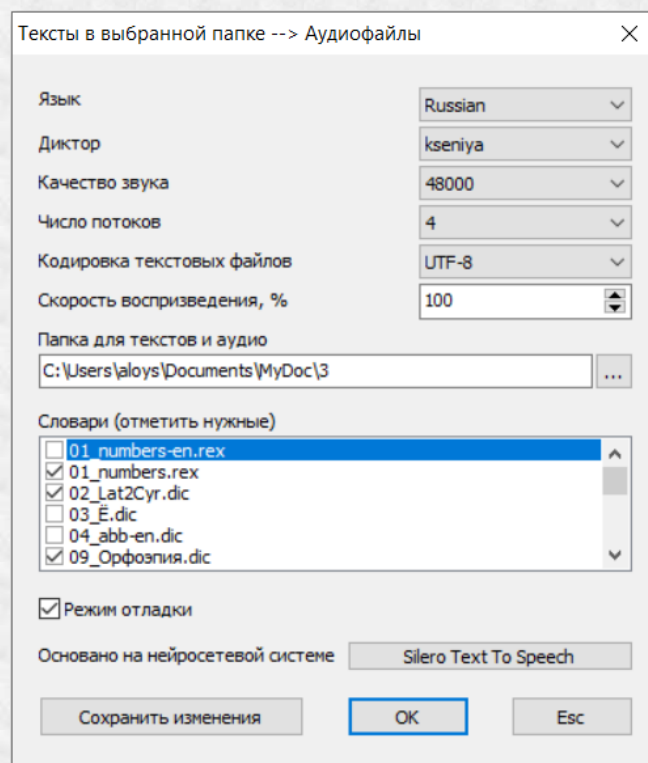




Озвучивается текст в активной вкладке Demagog. "Выполнить скрипт - Из файла Ctrl+F2". В открывшемся меню выбираем папку Silero, и запускаем скрипт Text to Speech.py
Тестировалось на ноутбуке MSI Katana 11th Gen Intel(R) Core(TM) i7-11800H @ 2.30GHz, RAM 16Gb. И на ноутбуке Acer Travel Mate B1 Intel (R) Celeron (R) N4120 CPU @ 1.10GHz, RAM 4Gb. На обоих устройствах установлена Windows 10 Домашняя 64bit.

Повесть Редьярда Киплинга "Отважные мореплаватели", 216 тыс. знаков = 6.4 а.л. - на MSI Катана озвучилась за 24 мин 06 сек. На Acer Travel Mate - примерно в 10 раз дольше. А если попадется книга потолще? В общем, для нейросетевого синтеза речи нужен достаточно мощный компьютер. Или придется запускать озвучку одной книжки на ночь, чтобы к утру получить результат.

Еще один скрипт, делает то же самое, но для всех текстовых файлов в заданной папке. Это могут быть, например, главы книги. Каждому тексту соответствует один аудиофайл.



!! Дистрибутив Demagog-x64-Silero при распаковке требует 1.4 Гб свободного места на жестком диске. 98% этого объема занимают нейросетевые модели для 4 языков, и модуль pytorch для работы с ними. Такая вот специфика. Требования к операционной системе: Windows 10, 11 - 64 bit.

Замечание. В августе 2023 г. Silero опубликовали модель синтеза речи V4, с ускоренной в 3-4 раза, как было заявлено, записью аудио. Фактически, по моим замерам, в 2 с небольшим раза. При этом наблюдается заметное ухудшение качества звука.

Есть, конечно, разница: 2 часа идет запись книжки идет или 50 мин? Но, если результат получается пьяно-хрипчатый или глухо-телефонный, то выигрыш по времени себя явно не оправдывает. Поэкспериментировав, по зрелому размышлению, я решил пока не добавлять в сборку Demagog-x64-Silero голоса V4. Как говорится: видно будет - посмотрим.

<<

19. Сам себе Гутенберг или электронная книга своими руками

- Ну что ж, мистер Сайрес, с чего начнем? - спросил на следующее утро Пенкроф.
- С самого начала, - ответил инженер.

Следуя героям известного романа, начнем с начала. В наш век интеллектуального прогресса каждый второй - писатель, фотограф и видеореporter. Всё, что для этого нужно - смартфон в руках. А вот объединить свои или кого-то другого репортажи в виде увлекательной книжки...

Э-э-э... хм-м... да уж... как-то слова начинают застревать в горле. А на самом деле, тоже ничего страшного. У вас есть хотя бы простенький ноутбук, хотя бы с Windows 7? Я вот извлек с дальней полки шкафа раритетный Emachines 350 с 1 Гб оперативной памяти. Эксперимент есть эксперимент.

С такими мыслями я поставил на эту слабую машинку программу Demagog. Если получится здесь, то и везде. Электронную книгу будем делать в формате epub - в настоящее время это самый популярный в мире формат. Для этого используем Lua-скрипт, включенный в состав Demagog, начиная с версии 402. Он представляет собой расширенную версию скрипта из сборки [Демагог одной кнопкой от tonio_k](#). Спасибо tonio_k за эффективно работающий скрипт и контрольный пример.

Идти по следам первопроходца легче, чем торить дорогу самому. Имея перед глазами рабочий образец, я сумел несколько расширить его функциональность. Теперь создаваемая электронная книга может содержать не только красиво отформатированный текст, но и:

- картинку - обложку;
- картинку - титульный лист;
- иллюстрации в заданных местах текста;
- список примечаний с кликабельными ссылками на них в тексте.

Начало работы. Каждая глава нашей книги должна представлять собой отдельный текстовый файл. Самый обычный текст - хоть в Блокноте набранный, хоть в Demagog или в другом вашем любимом текстовом редакторе.

Текстовые файлы надо называть так, чтобы их *алфавитный порядок* соответствовал порядку глав в книге. Первая строка каждой главы - *это ее название*. Если в книге несколько частей, то название каждой части - это как бы отдельная "глава", состоящая из одного лишь заголовка. Пример:

| Имя текстового файла | Что в нем находится |
|----------------------|--|
| _txt | Невероятная встреча на "южном материке" двух очень разных людей... |
| _.txt | Есть многое на свете, друг Горацио, что неизвестно нашим мудрецам. Вильям Шекспир |
| 100.txt | Часть I. Дружба народов |
| 101.txt | 1. СОСЕДИ Рассказывают, что были в Антарктиде две научные станции, принадлежавшие разным государствам - СССР и Чили. И находящиеся впритык друг к другу [1]. Настолько близко, что можно ходить друг другу в гости. Что обе стороны активно и делали с обоюдной выгодой. #i_001.jpg У чилийцев вечно техника ломается, сами понимаете, Антарктида - это вам не Южная Америка, здесь сильно прохладней. Чего чилийские производители тракторов и вездеходов не всегда учитывали. ... |
| 102.txt | 2. ТРАКТОР Еще бы не паниковать. Поломался самый мощный тягач. Без которого прямо беда - это если литературно выражаться. Руками, да и на санках грузы не очень-то потаскаешь. Морозы иной раз стоят такие, что собаки на поворотах ломаются (шутка, от которой не смешно). Так что Валера пришел в гости очень вовремя. ... |
| 200.txt | Часть II. Возвращение |
| 201.txt | 1. ДЕРЕВЯННЫЙ ЗАНАВЕС Антарктический контракт - год. Отработал, получил хорошие деньги и домой, отдыхать, впитывать тепло, любить женщин, скупко рассказывая очередной пассив о героических буднях прошедшей зимовки. Но, Антарктида манит. И через год Валера завербовался снова. И, представьте себе, его направили на ту же самую станцию. ... |
| 202.txt | 2. ЯХТА В один из таких мрачных дней Валера ... |
| 203.txt | 3. СКАНДАЛ Сладко потягиваясь от послеобеденного сна, тепло одетый, замполит вышел на крыльцо. И увидел ужасную картину ... |
| 300.txt | Примечания |
| 301.txt | 1 Были и есть. Старейшая в Антарктиде советская, ныне российская база Беллинсгаузен. Рядом официальный чилийский городок - Лас-Эстрельяс. Население 64 чел. Есть почта, отделение банка, школа, госпиталь. |

| | |
|---------|--|
| | отделение банка, школа, госпиталь. |
| 302.txt | 2 Несколько человек с таким именем и фамилией в разные годы работали в Антарктиде, но никто из них не подходит на роль героя нашей истории. Поэтому будем считать, что данный персонаж - это собирательный образ. |
| ... | ... |

Принцип понятен. Ссылки на примечания обозначаются цифрами в квадратных скобках. Текст каждого примечания - это маленькая "глава", с названием из цифры - номера примечания. Всем файлам примечаний предшествует файл с одним словом без точки на конце: Примечания

Положение картинки в тексте обозначается в отдельной строке как #filename. Пробелы и кириллица в именах картинок - запрещены! Допустимые типы иллюстраций: jpg, png. Файлы картинок обложки и титульного листа имеют предопределенные имена, соответственно: cover и title. Иллюстрации не являются обязательным элементом книги, хотя наличие обложки желательно.

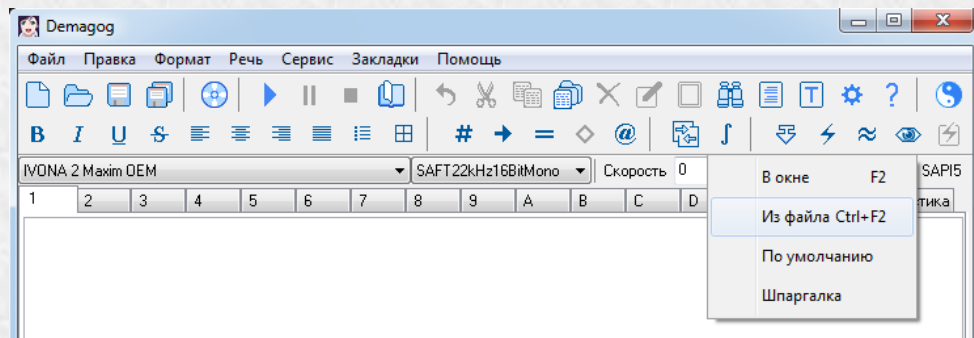
Допускается указывать стили текста в виде: <i>курсив</i> полужирный <i>полужирный курсив</i>. Имейте в виду, что соответствующий закрывающий тег должен находиться в том же абзаце, что и открывающий тег! Например, если хотите распространить стиль "курсив" на несколько абзацев, то каждый из них начинайте с <i> и завершайте </i>.

Если в книге есть аннотация, то ее файл имеет имя _.txt

Если в книге есть эпиграф, то его файл имеет имя __.txt

"И что дальше делать с этой кучей текстов и картинок?" Уместный вопрос. Положите всё это добро в какую-нибудь (изначально пустую!) папку на вашем компьютере. Это будет рабочая папка вашего издательского проекта.

Запуск скрипта. "Сервис - Статистика - Выполнить скрипт - из файла" или Ctrl+F2 или кнопка со знаком интеграла в Панели инструментов.



В появившемся меню выбираем: "E-books creation - TXT to EPUB Converter.lua". Открывается форма, вводим данные:

по русски

по английски

Автор

Н. Е. Известный

NoAuthor

Название

Случай в Антарктиде

Case in Antarctica

Краткий пересказ сюжета, 1-2 предложения

История невероятного знакомства двух очень разных людей

Универсальный уникальный идентификатор

https://www.uuidgenerator.net/

UUID

Рабочая папка с материалами проекта

C:\p00\5\

СОЗДАНИЕ ЭЛЕКТРОННОЙ КНИГИ В ФОРМАТЕ EPUB

Исходные тексты всех глав в формате txt должны находиться в отдельной папке. Желательно, чтобы размер каждой главы не превышал 200 кб. В той же папке должна

находиться по крайней мере 1 иллюстрация: обложка книги.

Важно!

Имена текстовых файлов должны быть такими, чтобы их алфавитный порядок соответствовал ПОРЯДКУ ГЛАВ в создаваемой книге! Например, для книги из 2-х частей: 100.txt - название первой части; 101.txt - глава 1; 102.txt - глава 2; ... 200.txt -

Очистить

OK

Esc

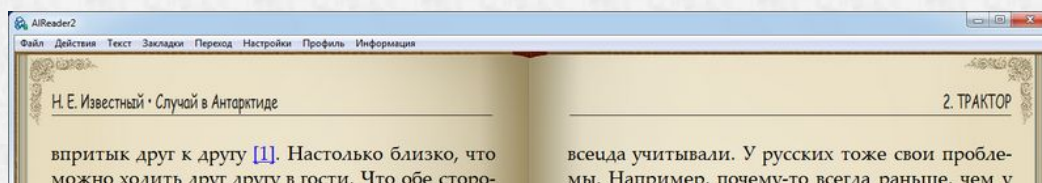
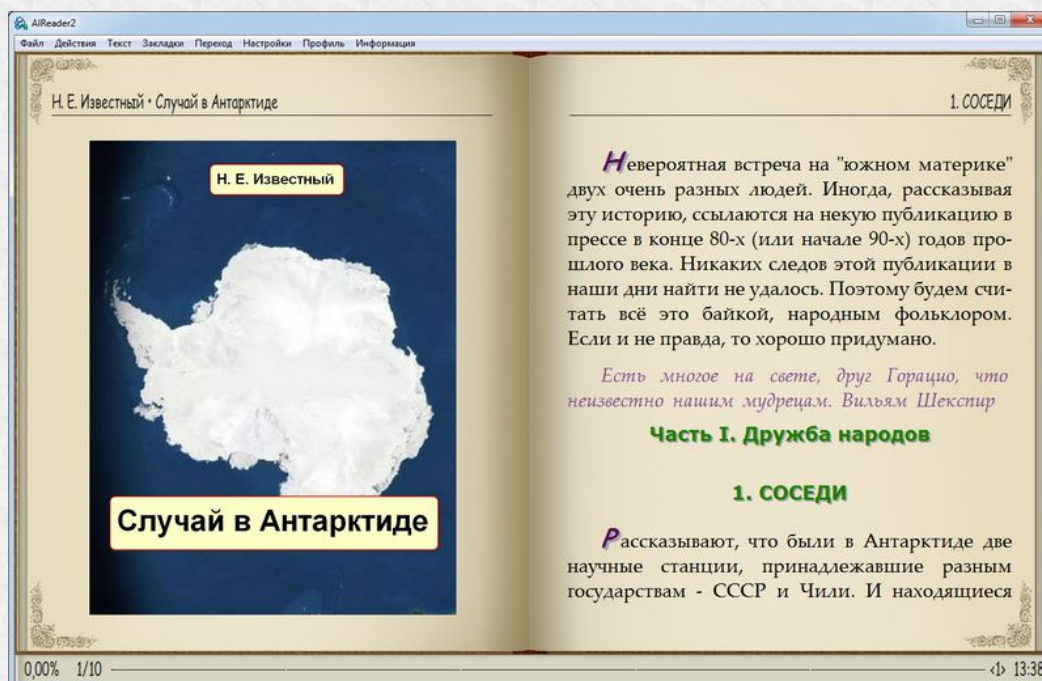
Обязательны для заполнения только английские название книги и имя автора - из них будет составлено имя файла книги. На крайний случай, если они-таки не заполнены, то название книги будет: new_book.epub.

Краткий пересказ сюжета - это **не** аннотация, а так называемый "логлайн", новомодное веяние. Некоторые программы-ридеры отображают его текст рядом с названием книги. Удобно, но не обязательно.

Уникальный идентификатор - бесплатная альтернатива ISBN. Если вы подключены к Интернету, то кнопка со ссылкой откроет сайт, где вы получите номер, которого больше ни у кого нет и не будет. На то он и уникальный. И опять же - нужный скорее для солидности.

Остается только указать, где лежат все материалы для вашей книги!

Потом давим кнопку "OK" и подносим к губам чашечку кофе или рюмочку чая... Допить не успеваем - уже готовая книга лежит в той же рабочей папке, дожидаясь благодарных читателей.



ны активно и делали с обоюдной выгодой.



У чилийцев вечно техника ломается, сами понимаете, Антарктида - это вам не Южная Америка, здесь сильно прохладней. Чего чилийские производители тракторов и вездеходов не

чилийцев, заканчивалось спиртное. Вот вам и основа для дружбы народов.

На советской станции работал в том далеком 1972 году механик Валера Сидоров [2]. Мастер на все руки, он стал частым и желанным гостем на "чилийской половине". В один из таких визитов он застал чилийских друзей в растерянности и волнении. И сразу понял: случилось что-то серьезное.

2. ТРАКТОР

Еще бы не паниковать. Поломался самый мощный тягач. Без которого прямо беда - это если литературно выражаться. Руками, да и на санках грузы не очень-то потаскаешь. Морозы

Если заметили - на титульном листе я сэкономил 😊 Кстати, для подготовки иллюстраций (изменение размера, обрезка, преобразование формата и т.п.) использовал бесплатную программу Fast Stone Image Viewer 7.5 Portable. Полный комплект файлов этого тестового проекта находится [здесь](#).

Готовые книги тестировал на программах-читалках: FBReader (десктоп), CoolReader (десктоп), AlReader (десктоп / андроид), EbookDroid (андроид), ReadEra (андроид), Moon Reader (андроид). А так же проверял на онлайн-ресурсе [EPUB Validator](#), ссылку на который мне прислал ув. tonio_k.

<<

20. Новые горизонты. 64-разрядная версия Demagog

- О, воины! Расскажите, как вырвались вы из вражьих застенков?

- Мы отважно сражались, но коварный враг пленил нас! И бросил в старый сарай. В котором мы сидели одну луну... Сидели две луны... Сидели три луны... А на четвертую... Зоркий Глаз заметил, что стен у сарая нет, только крыша на четырех столбах!..

Давняя история

Вот так и я однажды заметил... что 32-разрядные приложения устарели 😊 И Windows 10 выходит теперь только в 64-разрядном варианте. А из недр могучей корпорации, ее разработавшей, доносятся тонкие намеки на толстые обстоятельства. Мол, поддержка 32-разрядных приложений в режиме обратной совместимости пока есть, но... Прогресс неодолим, а кто сомневается, пусть пеняет на себя.

Пенять на себя я никак не захотел, и принял меры. В результате появилась новая линейка версий: Demagog-x64.

x64-версию имеет смысл использовать, когда на компьютере больше 4 Гб памяти. Впрочем, современные ПК и ноутбуки имеют как минимум 16 Гб ОЗУ. Грех этим не воспользоваться.

Есть, конечно и потери. SAPI4 в 64-разрядной версии не поддерживается, от слова "совсем". Настолько она древняя. В утешение скажу, что предыдущая линейка 32-разрядных версий Demagog по-прежнему будет мной сопровождаться. Так что, хриповатому старичку Николаю Елановичу Дигало найдется пристанище. На радость его, пусть уже и немногочисленным, фанатам.

А в новой x64-версии есть кое-что еще...

Интеграция с Python. Для этого я использовал библиотеку python4delphi, ее можно найти в Инете. Помещаем скачанные файлы библиотеки в какую-нибудь папку на нашем компьютере, например, P4D. Запускаем Delphi. Для Delphi 11.3 Alexandria установка компонентов выглядит так:

Project - Add Existing Project

..\P4D\Packages\Delphi\P4DComponentSuite.groupproj

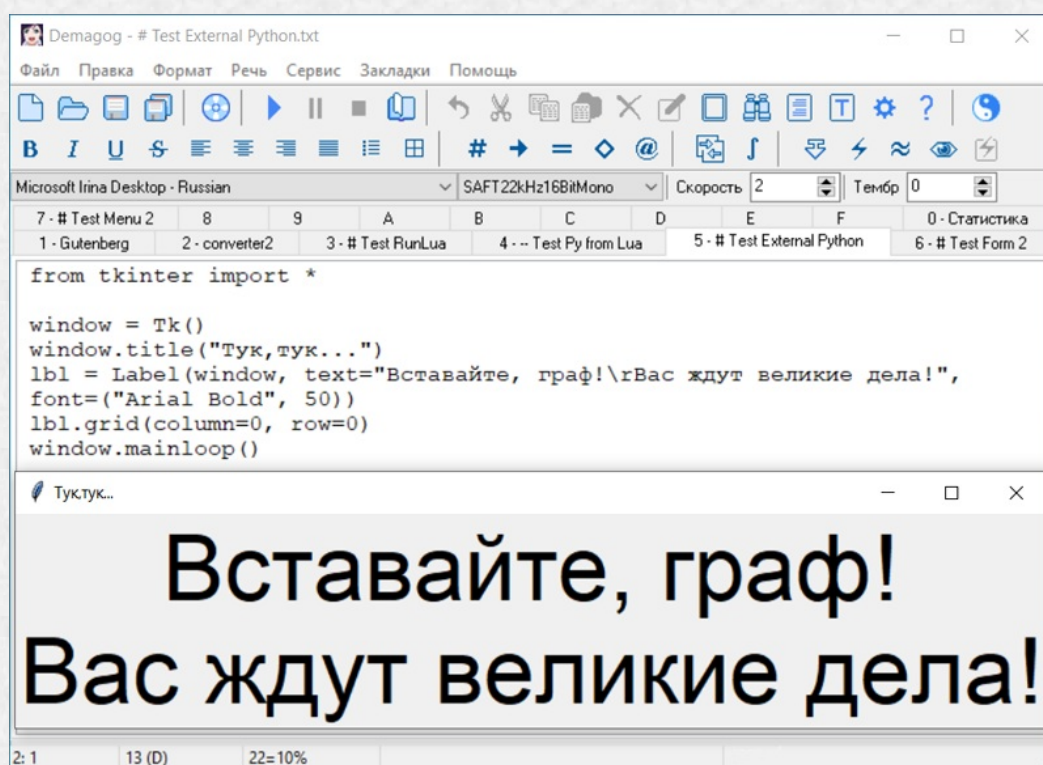
В появившемся дереве файлов для dclPython280.bpl, dclPythonVcl280.bpl, dclPythonFmx280.bpl поочередно правым кликом вызываем меню и выполняем Install.

Затем Tools - Options - Language - Delphi Options - Library, чтобы прописать пути к ..\P4D\Source, ..\P4D\Source\vcl, ..\P4D\Source\fmx.

В итоге, получаем набор компонентов, связывающих программу на Delphi с интерпретатором Python. Дельфийская прога сможет выполнять скрипты на питоне, и наоборот, некоторые ее функции можно, по желанию программиста, сделать составной частью питон-интерпретатора!

Скрипты на Python выполняются в Demagog аналогично скриптам на Lua, из того же меню. По умолчанию используется встроенный (embedded) питон-интерпретатор. Но, если на компьютере уже имеется установленный Python, то можно подключиться к нему. Для этого служит кнопка с изображением символа "инь-янь" или горячая клавиша F12.

Такая возможность полезна, когда работаем с Python, в который добавлены необходимые нам внешние библиотеки. Можно решать разнообразные задачи, выстраивая для них подходящий графический интерфейс.

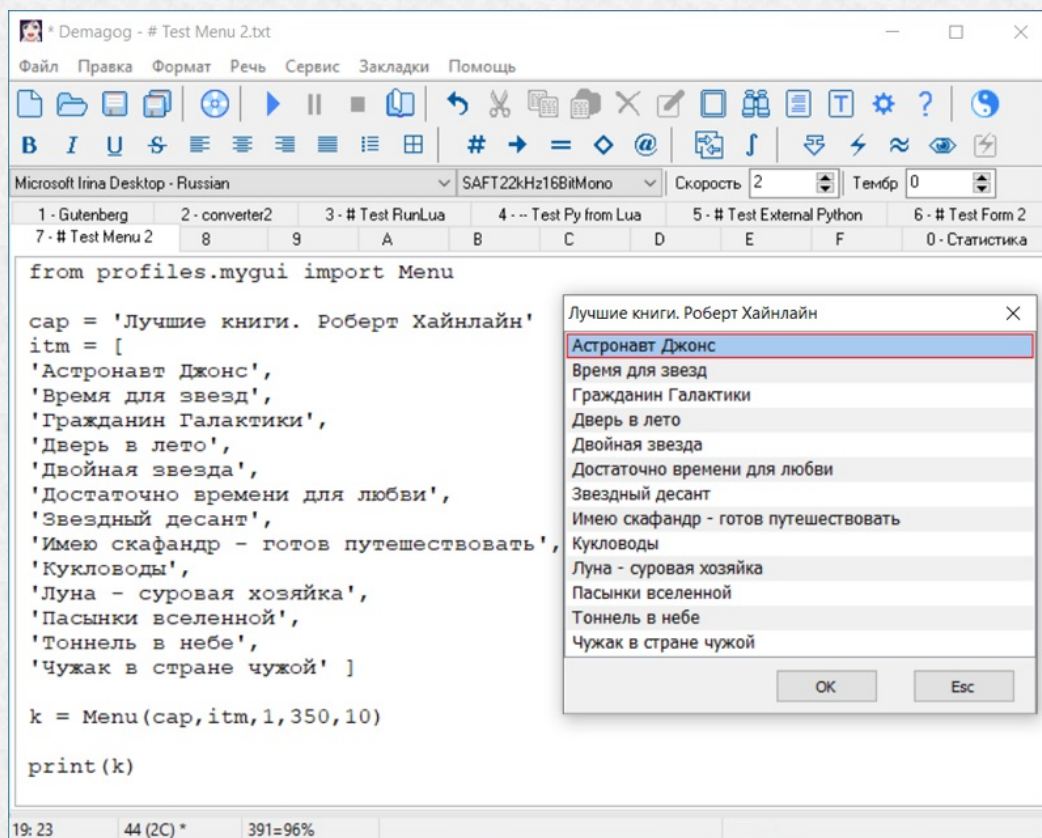


Важное замечание. *Встроенный (embedded) python не имеет в своем составе графической библиотеки.* При попытке выполнить на нем показанный на картинке пример - возникнет ошибка: No module named 'tkinter'. Более того, отсутствует и пакет pip, с помощью которого можно эту графику (или что-нибудь другое) установить. Такое вот, урезанное изделие. Ни убавить, ни тем более прибавить. Так решили его разработчики - принцип минимализма в действии. Так что, при использовании встроенным интерпретатором python может возникнуть некий дискомфорт.

На этот случай в x64-версии изначально предусмотрен для python доступ к отдельным элементам графического интерфейса Demagog. Минималистичный, скажем так, набор:

- Диалог открытия файлов (допускается множественный выбор) =>
- Диалог сохранения файла =>
- Показ окна сообщения =>
- Показ сообщения в строке состояния =>
- Индикатор "бегущая строка" =>
- Диалог обзора папок =>
- Диалог-меню =>
- Двух-колоночное окно ввода данных (вида "имя-значение") =>
- Универсальная форма ввода данных (для нее доступны 11 компонентов) =>

Пример использования вышеперечисленных функций приведен в папке `..\Demagog_Texts_samples python`. А отдельная демонстрация на рисунке ниже. По нажатию Shift + F2 простенький текст превратился в элегантное меню, шириной в 350 пикселей и размером шрифта 10.



Этого вполне достаточно для создания довольно развитого интерфейса. Еще один пример находится в конце [главы 18](#). С инструкцией можно ознакомиться в пункте главного меню Demagog "Сервис - Статистика - Выполнить скрипт - Шпаргалка".

<<

Заключение

На Demagog в Инете уже достаточно ссылок, в основном они ведут ко мне на страницу. Но есть и отдельно размещенные копии. Братья и сестры во Инете! Актуальный экземпляр Demagog со всеми причиндалами вроде аудио-кодеков, файлов справок и русской и английской орфотаблиц - скачивать нужно только с этой моей страницы.

Что говорят

24 Ролик на Ютуб: [Текст-в-голос | Demagog + Silero](#)

<https://nastroyvse.ru/programs/rating/luchshie-programmy-dlya-chteniya-teksta-golosom.html>

23 Программа объединила в себе функции текстового редактора, который поддерживает Юникод и синтезатор речи с поддержкой SAPI4/SAPI5. Важным отличием является то, что программа использует свои алгоритмы для словарных замен, которые помогают правильно произносить текст. Кроме того, в ней есть проверка орфографии и подсветка ошибок. Эта утилита может работать с различными текстами разных форматов MS Word, E-Book и HTML, даже может импортировать рисунки. В свою очередь, аудиофайлы могут быть также не только в формате MP3, но и WAV, OGG, WMA, MP4. Утилита работает с Windows XP, Vista, 7, 8, 10 (32/64 bits).

<https://vseprost.ru/top-programm-diya-ozvuchivaniya-teksta-golosom.html#Demagog>

Топ программ для озвучивания текста голосом

Проект Demagog – полезная утилита, которая больше походит на продвинутый текстовый редактор. Благодаря широким возможностям подходит не только для чтения и воспроизведения текстов, но и справляется с литературными произведениями. Порадует аудиалов следующими функциями:

- 22
- наличие инструментов форматирования, как в стандартных текстовых редакторах;
 - уникальный алгоритм корректировки произношения;
 - подсветка орфографических ошибок и омографов;

История версий

[+] добавления, [-] исправления/изъятия, [~] изменения/улучшения

| | |
|--|---|
| 7.30.422 7.30.422-x64 | <p>[~] Поправка в питон-скрипте синтеза речи на моделях Silero, для правильного отображения имен файлов, содержащих символы юникода.</p> <p>[~] Поправка в алгоритме запоминания списка ранее открытых файлов.</p> <p>[~] Улучшено извлечение текста из документов pdf. Текст извлекается, если документ имеет текстовый слой, за исключением т.н. "зашифрованных pdf".</p> <p>[~] Поправка в алгоритме извлечения текста из документов epub.</p> <p>[~] Поправка в алгоритме извлечения картинок из документов docx.</p> <p>[~] Поправки в опции "Найти/Заменить - Библиотекарь".</p> |
| 7.30.422-x64-Silero | <p>Экспериментальная сборка для чтения текстов голосами от компании Silero. Внимание! Этот дистрибутив, при распаковке, требует наличия 1.4 Гб свободного места на жестком диске. Кроме собственно Demagog, в нем содержатся нейросетевые модели для 6 языков: deutsch, english, french, russian, spanish, ukrainian. А так же модуль pytorch для работы с ними. Инструкция в pdf-формате находится внутри архива.</p> |
| | <p>[+] Добавлено извлечение текста из файлов: Adobe PDF (.pdf), MS Excel (.xls, .xlsx), MS Power Point (.ppt, .pptx), Open Document (.odt, .ods, .odp, .odg), OpenOffice.org (.sxw, .sxc, .sxi, .sxd).</p> <p><u>Замечание 1.</u> Из pdf-файла текст извлекается в том случае, если данный файл имеет текстовый слой. Кириллический текст извлекается корректно, если имеет юникодную кодировку.</p> <p><u>Замечание 2.</u> Иногда под видом формата .doc скрываются файлы .rtf или даже просто .txt. Теперь из таких "замаскированных" файлов текст извлекается корректно.</p> <p>[-] Исправлена ошибка извлечения картинок из документов doc, когда документ содержал</p> |