

Технические ограничения:

Внимание! *Программа распознаёт кодировку только по русским символам.*

Разработчики:

- **Сергеев Андрей** (*AndreyAs44*) - Логика основных алгоритмов, Внедрение и реализация алгоритмов в код, Программирование основного кода программы на Pascal, Рефакторинг кода, Помощь в организации рабочего процесса и ведении документации, Настройка и логика интерфейса программы.
- **Гаврилов Матвей** (*Kler1234*) - Поиск в источниках и разработка основных, различных идей и логики алгоритмов, Форматирование и обработка массивов данных на Python, Ревью кода, Ведение документации и таблиц, Тестирование программы на различные баги и важная обратная связь для улучшения проекта.
- **Яуров Глеб** (*Rendant*) - Ревью кода, Логика основных алгоритмов, Помощь реализации алгоритмов в код на Pascal, Обработка данных и создание матрицы частотности при помощи Python, Помощь в ведении документации и таблиц, Перевод и корректировка локализации (Английский язык)

Список доступных *различимых* кодировок:

Код страницы	Имя
437	DOS/IBM437
855	IBM855
865	NORDIC DOS IBM865
866	IBM CP866
950	Big5
1200	UTF-16LE
1201	UTF-16BE
1251	windows-1251
1252	windows-1252
10007	x-mac-cyrillic
20866	KOI8-R
21866	KOI8-U
28595	ISO-8859-5
65001	UTF-8

Список доступных команд:

- `locale` - смена языка в интерфейсе программы.
- `ИмяФайла.txt` - импорт текстового файла в программу.

Принцип работы программы:

1.Начало

Пользователь помещает файл с текстом в определённой кодировке в папку `In_Files`.

Выбирается язык интерфейса (`ru` или `en`). Вводится название файла `"ИмяФайла".txt`. Программа считывает текст и переводит в различные байтовые массивы.

2. Метод закономерных сортировок

Проверка на UTF-16

Программа начинает проверять байт-код с кодировки UTF-16 . Существует 2 вида кодировки UTF-16 : + UTF-16 LE + UTF-16 BE

Их разница заключается в том, что при выведении на экран байт-кода UTF-16 LE - первые значения выглядят следующим образом:

		"а"	"б"
255	254	48, 4	49, 4

А в UTF-16 BE - первые значения выглядят иначе:

		"а"	"б"
254	255	4, 48	4, 49

Т.е можно сделать вывод о том, что значения просто меняются местами.

Так как UTF-16 двух байтовая, поэтому в значениях "а" и "б" - выводится по два числа. А именно <символ в кодировке Unicode> и системный символ "0" или "4".
Благодаря системным символам (0 , 4), которых нет у других кодировок мы можем сделать вывод, что это одна из двух кодировок UTF-16

Если программе встречаются системные символы 0 и 4 , то начинается проверка в каком порядке они расположены. Если программа полностью подтвердила наличие всех значений - делается вывод кодировки в консоль и Log.txt

Проверка на UTF-8

Если проверка не обнаружила системных символов, то программа начинает сверяться с байт-кодом UTF-8. Мы берём другой (двойной) байт массив и сверяем значения, которые доступны только для UTF-8

Если в строке есть символ, который содержится в референсном байт-массиве UTF-8 , то строка относится к данной кодировке и выводится в консоль и Log.txt .

	UTF-8	Windows-1251
Буква	а	а
Двойной байт массив	1072	-3

4. Метод весов

upd: В новой версии программы добавляются баллы за частотность следующей буквы. Дополнение повышает точность определения некоторых кодировок. (например: буква "в" чаще встречается перед буквой "а")

Проверка KOI8-r, Windows-1251, DOS, OEM и другие

В том случае, если исходный файл не подходит под кодировки определяемым Методом закономерных сортировок , то запускается следующий блок проверки - Метод весов :

Основная идея метода весов, заключается в том, что функция проверяет символы в строке по порядку, и находит индекс символа в массиве проверяемой кодировки. Массив кодировки состоит из байт-кодов алфавита по порядку.

Далее программа начисляет баллы в зависимости от частотности буквы и частотности последующей. Данная функция повторяется для каждой кодировки.

По итогу, у нас получается массив с баллами для каждой кодировки. Кодировка набравшая наибольшее кол-во баллов побеждает и выводится пользователю.

Объяснение очень упрощено (например, весь алгоритм работает на байт-кодах)
Метод весов сверяет значение каждой буквы в тексте, смотрит какой index оно имеет и вычисляет сколько баллов (по частотности) нужно начислить данной кодировке.

К примеру, рассмотрим баллы с учётом частотности буквы:

Буквы	а	б	в	г
Частотность	8.01	1.59	4.54	1.70

Буквы	а	б	в	г
CP866	160	161	162	163
IBM855	160	162	235	175

У каждых из кодировок буквы находятся в уникальном месте таблицы `Unicode` . Программа считывает каждую букву в *исходном* тексте: цикл встречает букву `a (index[0])` со значением в байт-массиве `160` . Значение `160` есть только в 1 кодировке - это CP866. Затем, программа сравнивает значение в массиве частотности `index[0]` и добавляет к общему *"весу"* кодировки CP866 значение **8.01**

```
Например, слово "авва" (160 162 162 160).

CP866 = 8.01(160) + 4.54(162) + 4.54(162) + 8.01(160) = 25,1
IBM855 = 8.01(160) + 1.59(162) + 1.59(162) + 8.01(160) = 19,2

Кодировка CP866 набрала больше баллов и, следовательно, победила.

Данный метод напрямую зависит от количества слов в тексте. Чем текст больше, тем выше точность алгоритма.
```

И по такому же принципу проверяются все остальные буквы. Затем просматривается какая кодировка заняла больший *"вес"* - она же передаётся в консоль и `Log.txt`

На этом программа завершается. Если ни одна кодировка не определилась, то программа выведет ошибку `Кодировка либо символы не поддерживаются данной программой` .

Для разработчиков

Программа для определения байт-кода (GetBytesFromFile)

Так же для тех, кто хочет добавить в программу свою кодировку есть специальная утилита (находящаяся по пути -> `..\Tools`). В папку `In_files` помещается (-ются) файл (-ы) с алфавитом (*абвгдеёжзийклмнопрстуфхцчшщъыьэюяАБВГДЕЁЖЗИЙКЛМНОПРСТУФХЦЧШЩЪЫЬЭЮЯ*) в нужной для вас кодировке. Программа возвращает значения в байт-коде, который впоследствии вы можете интегрировать в программу.

```
Внимание! Не все кодировки имеют совместимость с методом весов. Если программа не выдаёт нужный результат - попробуйте определять другим методом.
```