Российская Федерация

Ханты-Мансийский автономный округ – Югра

Департамент образования и науки

Сургутский государственный университет ХМАО

Политехнический институт

Кафедра автоматики и компьютерных систем

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА К КУРСОВОМУ ПРОЕКТУ

По дисциплине «Программирование и основы алгоритмизации»

Выполнила: студентка группы 609-11

Пашина И. Р.

Проверил: доцент кафедры

Автоматики и компьютерных систем

Гришмановский П. В.

Сургут

2023 г.

# **Аннотация**

Данный курсовой проект по программированию посвящен изучению и практическому применению Азбуки Морзе. В проекте рассматриваются основные принципы кодирования и декодирования сообщений в Азбуке Морзе.

Целью проекта является предоставление полного обзора Азбуки Морзе в контексте программирования, развитие навыков работы с кодированием и декодированием сообщений.

В главе «Анализ» рассматривается предметная область, обзор существующих решений, на основании которых строятся требования к разрабатываемому продукту.

В главе «Проектирование» строится поведенческая модель разрабатываемого ПП, представление данных и основные алгоритмы.

В главе «Кодирование» представлена программная реализация кода с описанием каждой функции.

В главе «Тестирование» проведены тесты для проверки работоспособности программы в различных ситуациях.

В главе «Сопровождение» представлена инструкция по эксплуатации программы.

Получившийся программный код в ходе курсового проекта представлен в «Приложении А»

# **Задание**

**Азбука Морзе.**

Программа предназначена для перевода текста на русском и английском языках в код азбуки Морзе и обратно. Исходный текст и результат перевода могут быть представлены как в файле, так и при помощи консоли.

Самостоятельно определить в процессе работы и согласовать с руководителем следующие требования:

- формат входного и выходного файлов;

- способ записи азбуки Морзе в виде текста с использованием стандартных символов таблицы ASCII.

# **Содержание**

[**Аннотация** 2](#_Toc147879199)

[**Задание** 3](#_Toc147879200)

[**Содержание** 4](#_Toc147879201)

[**Введение** 5](#_Toc147879202)

[**Анализ** 6](#_Toc147879203)

[**Проектирование** 11](#_Toc147879204)

[**Кодирование** 15](#_Toc147879205)

[**Тестирование** 22](#_Toc147879206)

[**Сопровождение** 26](#_Toc147879207)

[**Заключение** 27](#_Toc147879208)

[**Литература** 28](#_Toc147879209)

[**Приложение А** 29](#_Toc147879210)

# **Введение**

Азбука Морзе является увлекательной и актуальной для изучения и понимания системы передачи информации с помощью звуковых и световых сигналов. Азбука Морзе, разработанная Сэмюэлем Финли Бреселом Морзе и Альфредом Вэйландом, была одной из первых эффективных систем связи на расстоянии и использовалась для передачи сообщений по телеграфу.

В данном курсовом проекте мы познакомимся поближе с азбукой Морзе, со способом её кодирования, разработаем свою программу, которая позволит автоматизировать процесс перевода и поможет не опытным пользователям понимать Морзе.

# **Анализ**

**Описание предметной области**

Азбука Мо́рзе [1], «Морзя́нка», Код Мо́рзе — способ знакового кодирования, в котором буквы алфавита, цифры, знаки препинания и другие символы представляются в виде последовательностей длинных и коротких сигналов, называемых точками и тире (рис.1.). Предназначена для передачи по последовательным каналам связи. Уникальной особенностью азбуки Морзе является возможность кодирования и декодирования человеком без применения специальных терминальных устройств.

Наиболее широко используется слуховой прием азбуки Морзе, который получил распространение в радиосвязи (слуховой радиотелеграф). В ВМФ код Морзе применяется в световой связи между кораблями, осуществляемой при помощи специальных сигнальных прожекторов. Тактильная передача кода Морзе встречается редко, в частности, она есть в некоторых моделях смарт-часов.

В стандартном коде Морзе за единицу времени принимается длительность самого короткого сигнала — точки. Длительность тире равна трём точкам. Пауза между элементами одного знака — одна точка, между знаками в слове — 3 точки, между словами — 7 точек. Код может передаваться с любой доступной скоростью, причем возможность декодирования сохраняется и при значительных неточностях в соблюдении временных интервалов.

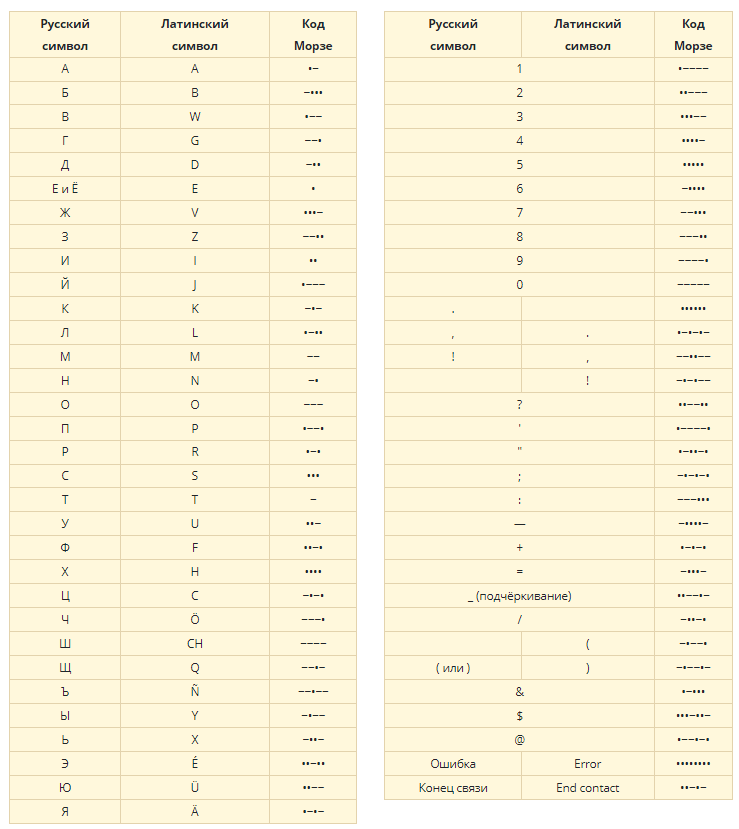


Рисунок.1. таблица перевода в азбуку Морзе

Суть программы заключается в автоматическом переводе слов в азбуку Морзе и обратно. В программе будет доступно два языка для перевода: русский и английский.

**Обзор существующих решений**

Программы-аналоги:

1. Онлайн калькулятор: Переводчик азбуки Морзе [2] (рис. 2.).

Принцип работы: Пользователь загружает текст, требующий перевода (Текст в код Морзе, Код Морзе в кириллицу, Код Морзе в латиницу). Программа начинает сопоставлять заданные буквы с кодом Морзе и переводит текст посимвольно.

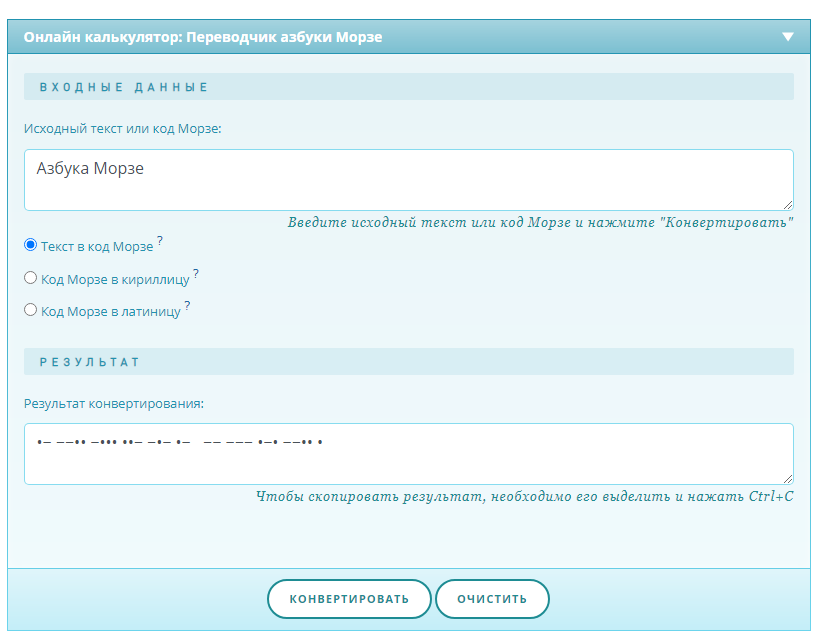


Рисунок 2 - Пример работы программы «Онлайн калькулятор:

Переводчик азбуки Морзе»

1. Перевод из одной системы счисления в другую [3] (рис. 3.).

Принцип работы: Перевод целого положительного числа в систему счисления с другим основанием происходит путём деления на основание. Полученное частное снова делиться на основание, и дальше до тех пор, пока частное не окажется меньше основания. В результате полученное число записывается в одну строку (последнее частное и все остатки, начиная с последнего).

Пример работы:

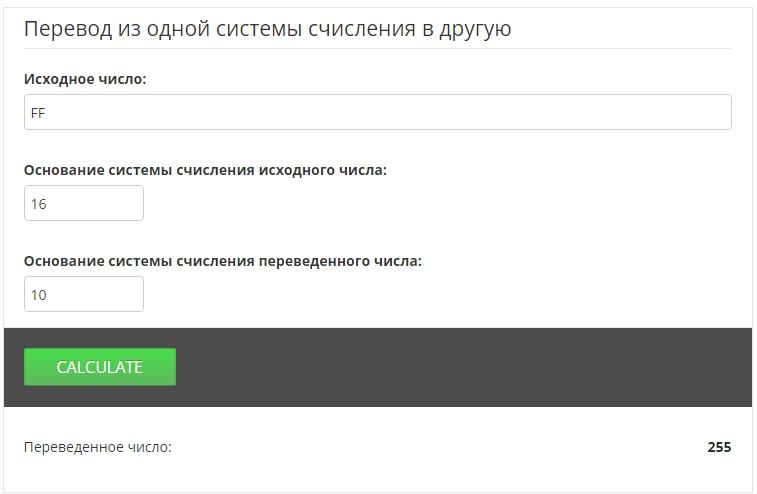


Рисунок. 3. Пример работы программы «Перевод из

одной системы счисления в другую»

3)Яндекс переводчик [4] (рис. 4.).

Принцип работы: Пользователь загружает текст, требующий перевода на конкретный язык, программа начинает сопоставление введенных слов, загруженных в ее словарь, и переводит текст дословно. После сравнения и замены каждой словесной единицы перед пользователем представят «переведенный текст».

Пример работы:

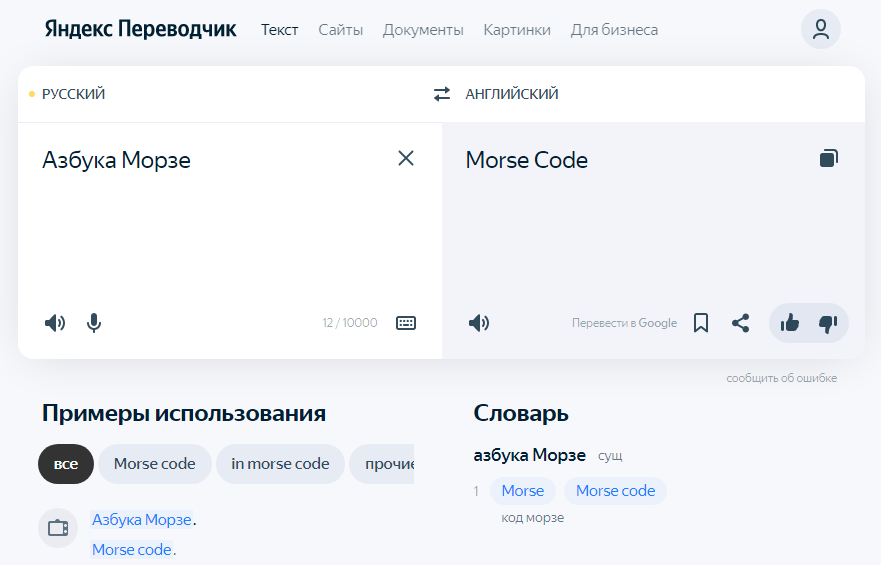


Рисунок 4. Пример работы программы: «Яндекс переводчик»

**Требования к разрабатываемой программе**

Разрабатываемая программа должна содержать:

1. Посимвольный перевод.
2. Перевод более одного слова.
3. Организованное меню с выбором перевода (с русского (английского) в азбуку Морзе и обратно).
4. После каждого сеанса запрашивать, хочет ли пользователь продолжить использование программы (в случае согласия – запускаем программу снова, в случае отказа – выходим из программы).
5. Возможность вводить слова при помощи консоли или загрузки файла.
6. Неизвестный символ в коде Морзе на «#»
7. Неизвестны символ в тексте заменить на «#»
8. Организовать сохранение форматирования при загрузке файла

# **Проектирование**

**Поведенческая модель программного продукта**

Для взаимодействия с пользователем будет выводиться меню:

1. Перевести текст с русского в код азбуки Морзе
2. Перевести текст с английского в код азбуки Морзе
3. Перевести код азбуки Морзе на русский
4. Перевести код азбуки Морзе на английский
5. Очистить экран
6. Завершить

В первых четырёх пунктах меню будет предложено загрузить/сохранить текст файла.

**Структура файлов, используемых и/или формируемых программой**

Программа будет работать с файлами типа txt.

**Представление данных в памяти при их обработке.**

Поскольку перевод производиться побуквенно и комбинации точек и тире для русских и латинских символов одинаковы, но используются для разных символов, то было решено использовать массив структур. Структура будет состоять из 3-х переменных: русский символ, латинский символ, строка в коде Морзе.

**Основные алгоритмы, соответствующие целевым функциям программы, с пояснениями к ним, необходимыми для их понимания**

**Алгоритм ввода строки произвольной длины (рис. 5)**

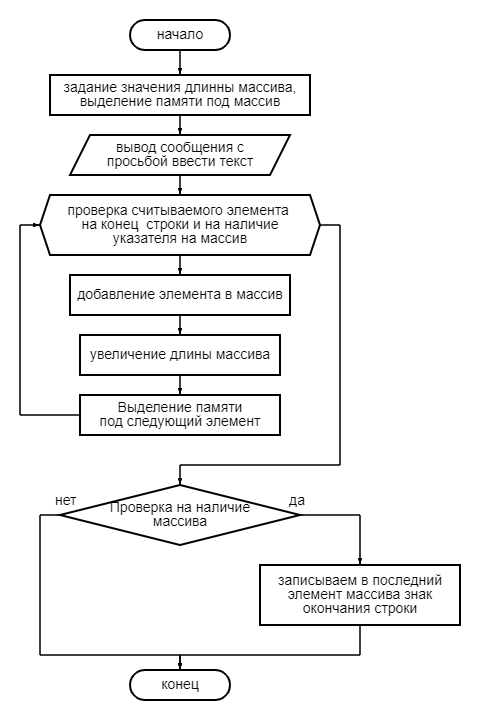


Рисунок. 5. Алгоритм ввода строки произвольной длины

**Алгоритм разделения строки кода Морзе и подсчёт комбинаций (рис. 6)**

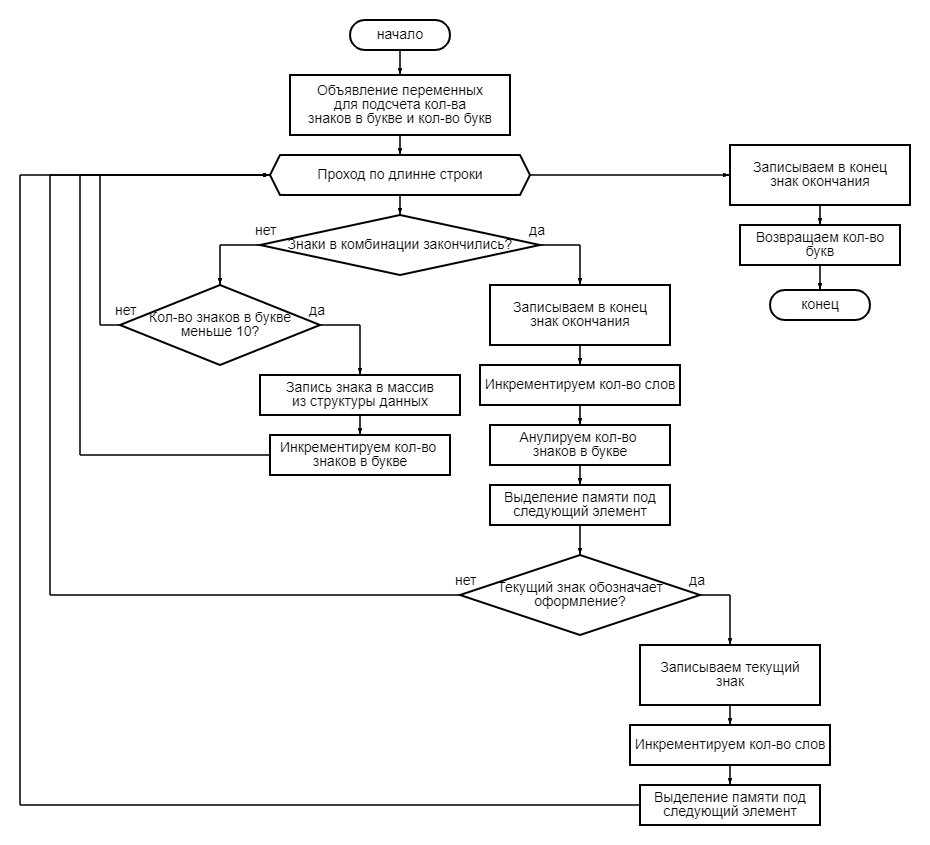


Рисунок 6. Алгоритм разделения строки кода Морзе и подсчёт комбинаций

**Алгоритм перевода (рис. 7)**

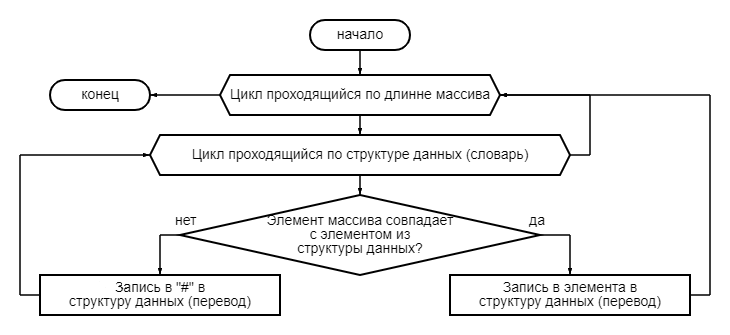


Рисунок 7. Алгоритм перевода

# **Кодирование**

Для разработки программного продукта был выбран язык С по стандарту ISO C99 в среде «Dev-C++ 5.11», поскольку именно этот язык был пройден в течении курса «Программирование и основы алгоритмизации».

После анализа и проектирования были сформулированы следующие задачи:

1. Составить словарь по рисунку 1;
2. Организовать меню;
3. Написать функцию для считывания строки произвольной длины;
4. Написать функцию для перевода с русского языка в азбуку Морзе;
5. Написать функцию для перевода с английского языка в азбуку Морзе;
6. Написать функцию разделения азбуки Морзе на буквы и подсчёта их количества.
7. Написать функцию для перевода с азбуки Морзе на русский язык;
8. Написать функцию для перевода с азбуки Морзе на английский язык;
9. Написать функцию для загрузки файла;
10. Написать функцию для сохранения файла;

Были подключены заголовочные файлы «stdlib.h», «stdio.h» и «string.h»

«stdlib.h» - данная библиотека нужна для использования динамической памяти

«stdio.h» - данная библиотека позволяет работать с файлами, считывать вводимые данные и выводить текст.

«string.h» - Данная библиотека позволяет копировать строки.

Для создания словаря было решено использовать массив структур. Каждая структура содержит букву русского языка, букву английского и комбинацию Морзе (рис. 8). Для букв русского и английского языка используется тип данных char, а для Морзе – массив данных типа char. Размер массива был рассчитан следующим образом: максимальная длинна комбинации (8 символов) + символ окончания строки (‘\0’) + 1 символ.

Входные данные могут быть непредсказуемы. Они могут быть как существующими, так и не существующими. Существующие приведены на рисунке 1. Максимальная длина имеющихся комбинаций, как уже говорилось, имеет 8 знаков, но на вход может податься комбинация содержащая большее количество знаков. Если бы в структуре массив Morse содержал именно 8 элементов + ‘\0’, то появляется вероятность, что комбинацию большей длины записали бы отсечённой и смогли перевести. Чтобы такого не происходило было решено добавить ещё один знак в массив. Это позволит нам решить проблему с неправильным переводом.

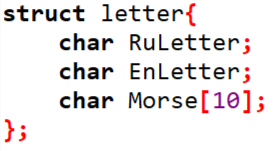


Рисунок 8. Структура данных для словаря.

Пример реализации словаря приведён на рисунке 9. Поскольку язык программирования чувствителен к регистру, часть структур была продублирована для возможности перевода нижнего и верхнего регистра.

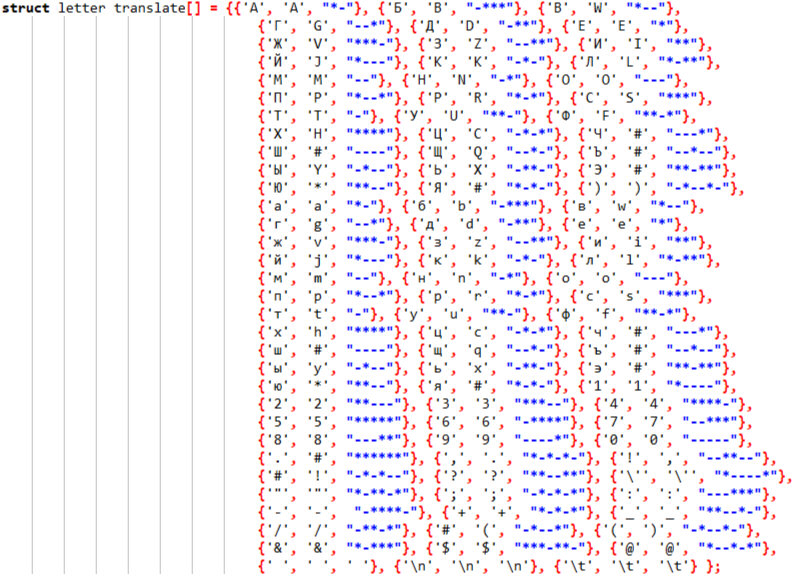


Рисунок 9. Словарь.

Было создано меню (рисунок 10). Первые 4 пункта позволяют осуществить перевод, 5 пункт очищает экран, а 6 – выход из программы.

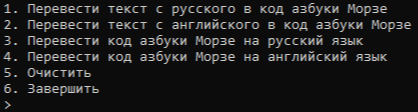


Рисунок 10. Меню.

При выборе одного из переводов будет предложено загрузить файл (рис. 11). Если пользователь выбрал загрузить файл, то будет вызвана функция «LoadFile» (рис. 13). В противном случае будет вызвана функция «EnteringString».

Следующим шагом будет проверка на наличие массива строк. Если массив не пуст, то происходит выделение памяти под массив структур, перевод и вывод на экран. Также будет предложено сохранить перевод в файл.

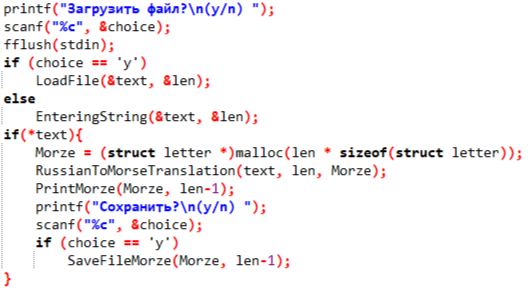


Рисунок 11. Принцип работы пункта перевода с русского на Морзе

Были реализованы функции для ввода (рис. 12) и загрузки файла (рис. 13). Так как функция загрузки файла была основана на функции ввода (алгоритм представлен на рисунке 5), сначала рассмотрим её.

Функция принимает 2 параметра: Указатель на указатель (указатель на массив) типа char «t», в котором будет храниться строка, и указатель «len» типа int, которая будет содержать длину строки (массива). Также в данной функции была создана переменная «с» типа char для считывания 1 вводимого символа.

Значения переменной «len» присваивается 1 и выделяется память под 1 элемент массива.

Далее идёт цикл while. Был выбран именно этот цикл из-за отсутствия счётчика, что позволит считать строку произвольной длины, не потеряв ни одного элемента. Условия выхода из цикла:

1. Достигнут конец строки
2. Массив символов пуст

Тело цикла содержит:

1. Запись считанного элемента в массив
2. Инкрементирование длины
3. Перевыделение памяти

Когда достигнут конец строки, происходит выход из цикла. Так как в цикле есть 2 условия выхода, то написана повторная проверка на то, что выход был произведён не из-за ошибки выделения памяти под массив, тогда в последний элемент массива записывается знак окончания строки «\0».

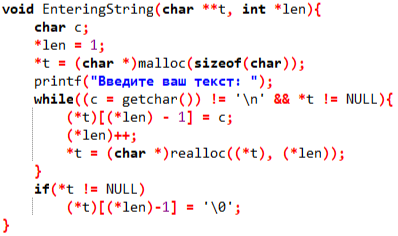


Рисунок 12. Функции ввода вручную

Функция «LoadFile» аналогична «EnteringString», за исключением того, что создаются 2 новые переменные: 1 – указатель на файл и 2 – название файла (массив типа char длиной 255 символов (максимальная длина названия файла)).

Данная функция запрашивает название файла, после чего идёт попытка открытия файла для чтения. Если не получилось открыть файл, то выводиться соответствующее сообщение об ошибке. В случае успеха мы попадаем в такой же цикл, что и в предыдущей функции. Только считываемый элемент проверяется на конец файла.

В конце функции обрабатываемый файл закрывается.

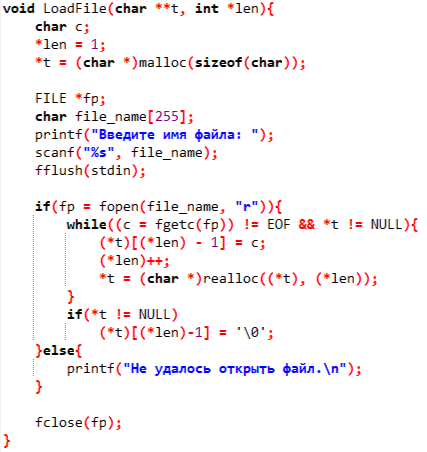


Рисунок 13. Функция загрузки файла.

Функции перевода в азбуку Морзе идентичны, поэтому рассмотрим одну из них (Алгоритм представлен на рисунке 7). Функция «RussianToMorseTranslation» (рис. 14) принимает 3 аргумента:

1. Указатель на строку
2. Длина строки
3. Массив структур

Перевод происходит благодаря циклу в цикле. Первый цикл проходиться по строке, а второй по словарю. В теле второго цикла происходит сравнение элемента строки с элементом словаря. Если элементы совпали, то в структуру записываются введённый знак и перевод. Если в словаре не нашлось подходящего знака, то в переводе он заменяется на «#».

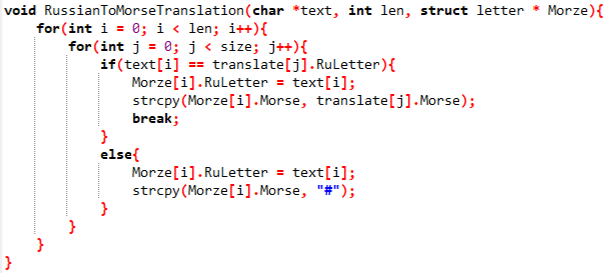


Рисунок 14. Функция перевода с русского языка в азбуку Морзе.

Обратный перевод выполняется сложнее, поскольку сохранение исходной строки происходит в массив типа char, появляется необходимость разбить строку на комбинации, а также сосчитать их для выделения необходимого количества памяти. (Алгоритм функции рисунка 15 представлен на рисунке 6). Поэтому перевод в Морзе осуществляется в два этапа:

1. Разбиение строки, и подсчёт комбинаций
2. Перевод

В функции CountAndSplitLettersMorse (рисунок 15) предусмотрено избежание переполнения массива Morse, поскольку стоит ограничение в 10 символов, то оставшаяся часть комбинации будет отсечена. Также она учитывает разделение между комбинациями (1 пробел), составляющие 1 слово. Этот пробел будет поглощён и слово будет записано целиком, без пробелов. Но если это разделение между словами (от 2 пробелов), то последующие пробелы будут записаны в структуру в виде знака окончания строки, которые в последующем будут расшифрованы в функции MorseToRussianTranslation (рис. 16). Помимо этого, функция способна различать абзацы и табуляцию при помощи дополнительной проверки.

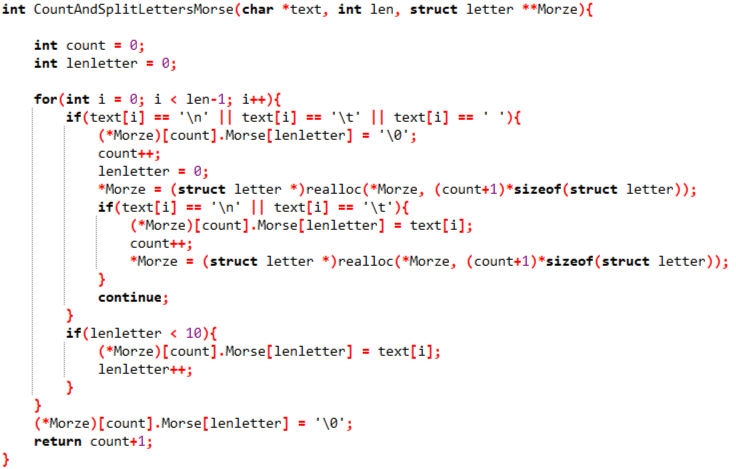


Рисунок 15. Функция разбиения строки Морзе на комбинации и их подсчёт.

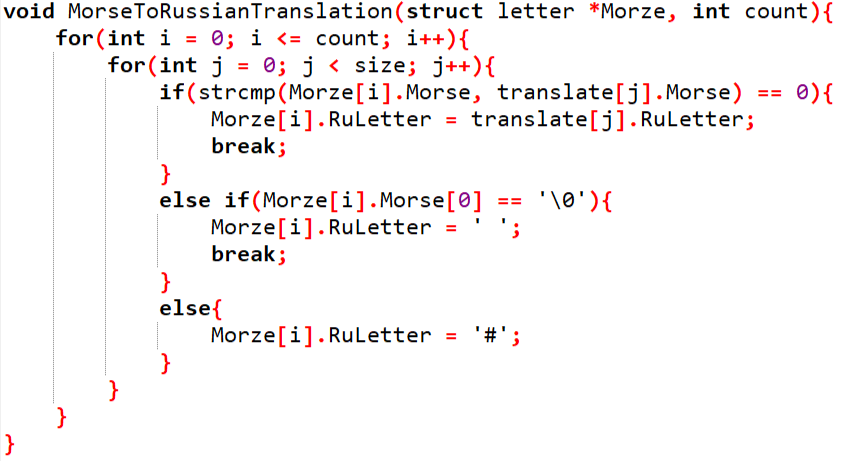


Рисунок 16. Функция перевода с Морзе на русский

Сохранение перевода в файл происходит при помощи функций SaveFileRu, SaveFileEn, SaveFileMorze. Они однотипные, поэтому рассмотрим одну из них. Функция, приведённая на рисунке 17, работает также как функция загрузки файла (рис. 13), но вместо считывания содержимого происходит запись в файл.

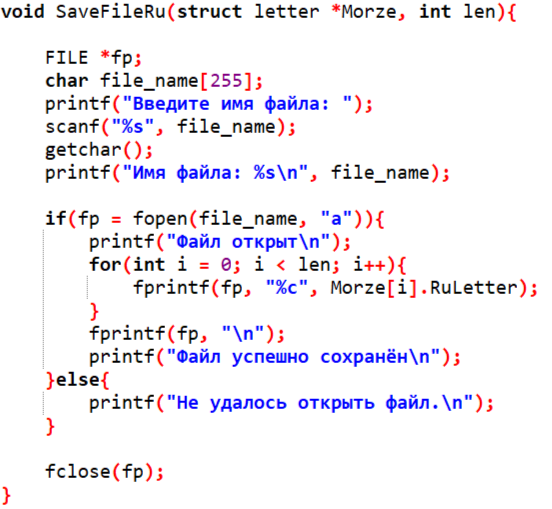


Рисунок 17. Функция сохранения.

# **Тестирование**

Чтобы проверить устойчивость и адекватность программы проведём тестирование:

1. Перевод с русского и обратно;
2. Перевод с английского и обратно;
3. Программа встретила неопознанный знак;
4. Программа встретила несуществующую комбинацию в азбуке Морзе;
5. Проверка загрузки и сохранения файла.

**1. Перевод с русского и обратно**

Для проведения эксперимента возьмём случайно сгенерированный текст [5]:

«Практический опыт показывает, что постоянное информационно-техническое обеспечение нашей деятельности требует от нас анализа соответствующих условий активизации. Не следует, однако, забывать о том, что постоянное информационно-техническое обеспечение нашей деятельности обеспечивает актуальность форм воздействия! Не следует, однако, забывать о том, что рамки и место обучения кадров играет важную роль в формировании соответствующих условий активизации.»

На рисунке 18 приведён перевод текста выше. Чтобы проверить правильность перевода, выполним обратный перевод. (Поскольку память при выборе каждого пункта меню выделяется заново, это служит гарантом того, что текст будет именно переведён, а не сохранён в памяти и выведен)

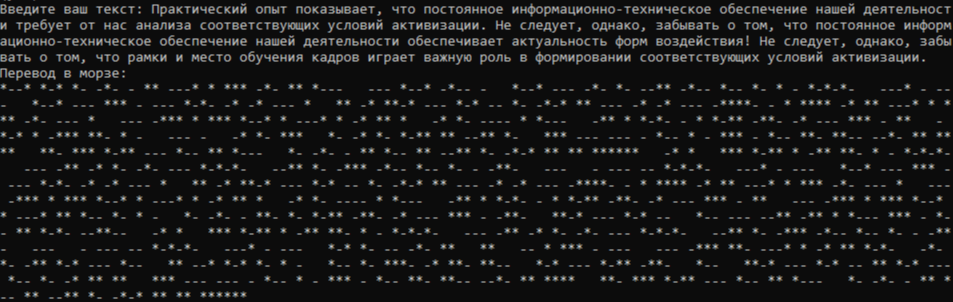


Рисунок 18. Перевод с русского в Морзе.

Пример обратного перевода изображен на рисунке 19. Как видим содержание осталось прежним, поэтому программа работает исправно. Следует отметить, что обратный перевод не может отвечать за регистр, поэтому принято использовать заглавные буквы (это не ошибка). Программа отработала корректно.

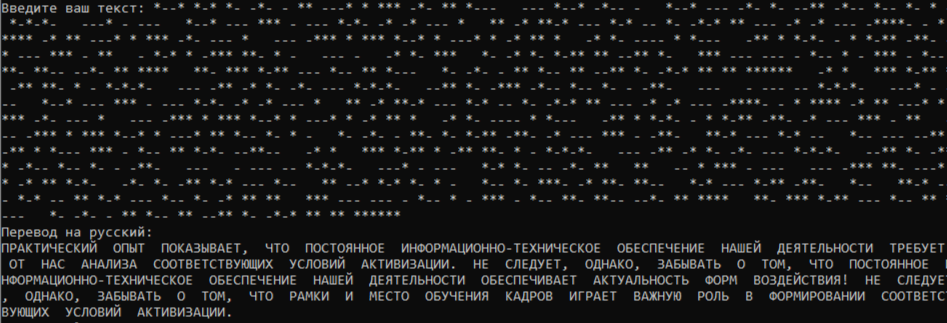


Рисунок 19. Перевод с Морзе на русский.

**2. Перевод с английского и обратно**

По аналогии сгенерируем текст [6] и проверим ПП:

«Civilly hoped sold letter total folly unable removal come throwing motionless allow agreement reserved. Admitting believing spot vicinity day wonder inhabit formed begin. Unlocked husband pleasant carriage passed park whose demands sure kindness ladies cousins shed may away roused furniture. Woman great warrant too mile mother over. Water ecstatic way called earnestly full sake denote performed justice announcing newspaper between neglected window my match. »

На рисунках 20 и 21 приведены примеры работы программы. Как и в случае с русским языком, всё отработало корректно.

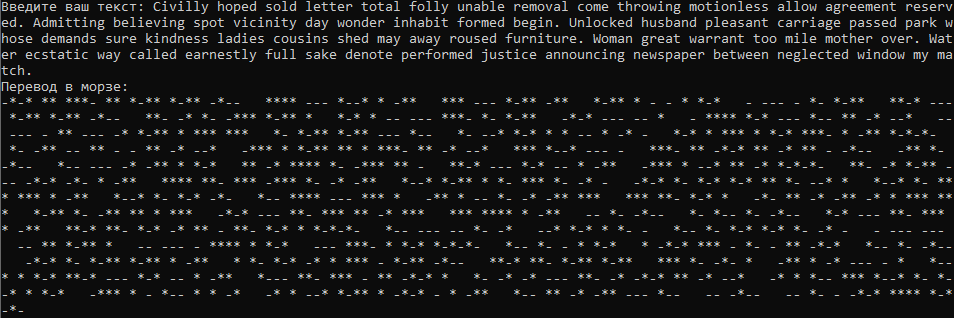


Рисунок 20. Перевод с английского в Морзе.

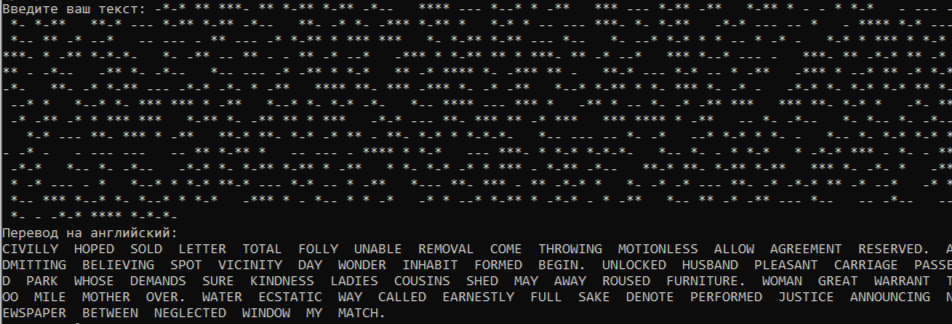


Рисунок 21. Перевод с Морзе на английский

**3. Программа встретила неопознанный знак**

Возьмём за неопознанный знак слово из другого языка. Каждую неопознанную букву программа должна заменить на символ «#». Возьмём по предложению из предыдущих пунктов и вставим слово противоположного языка.

Для английского: «Civilly hoped опыт letter total folly unable removal come throwing motionless allow agreement reserved».

Для русского: «Практический sold показывает, что постоянное информационно-техническое обеспечение нашей деятельности требует от нас анализа.».

Как видим, на рисунках 22 и 23 мы получили ожидаемый результат. Значит ПП справляется с обработкой данных ошибок корректно.

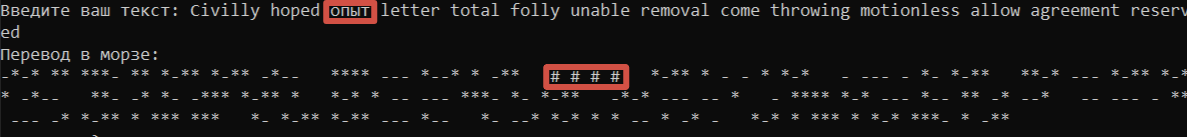


Рисунок 22. Обработка неизвестного знака при переводе с английского.

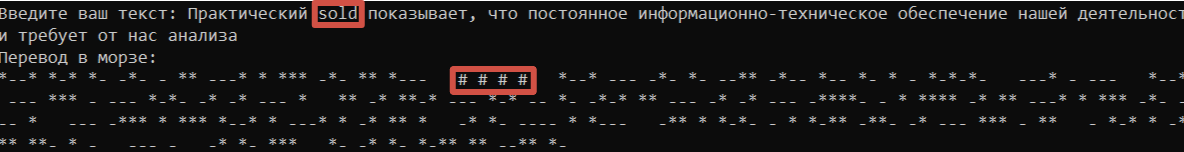


Рисунок 23. Обработка неизвестного знака при переводе с русского.

**4. Программа встретила несуществующую комбинацию в азбуке Морзе.**

Проведём следующие тесты теста:

1. Комбинация от 8 символов (\*\*\*\*----\*\*\*--)
2. Существующая комбинация со знаком одного из алфавитов (-\*\*\*Б)
3. Не существующая комбинация до 8 символов (\*\*\*\*\*\*\*)

Во всех случаях ожидается, что программа не сможет перевести и заменит комбинацию на «#»

На рисунках 24, 25 и 26 приведены примеры работы программы. Программа отработала корректно.

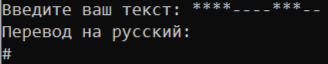


Рисунок 24. Обработка несуществующей комбинации более 8 знаков

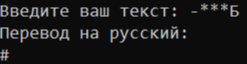


Рисунок 25. Обработка существующей комбинации с буквой



Рисунок 26. Обработка несуществующей комбинации до 8 знаков

**5. Проверка загрузки и сохранения файла**

Так как все 4 пункта меню на перевод работают одинаково, то нет смысла рассматривать все. Достаточно рассмотреть 1.

Для проверки программного продукта используем тест: «1234 56789». Сначала попробуем его записать в файл, а после считать. Результат выполнения программы представлен на рисунке 27.

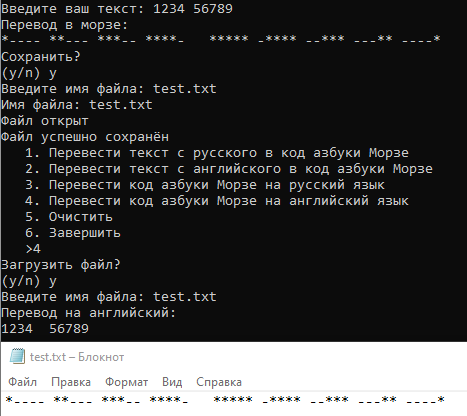


Рисунок 27. Пример записи и считывания текста

# **Сопровождение**

Данный продукт будет распространятся в виде исполняемого файла. Файл может запускаться на компьютерах/ноутбуках с операционной системой Windows 10 х64.

Размер файла: 168 Кб.

**Инструкция для пользователя**

На рисунке 28 приведено меню, с которым встречается пользователь, запустивший ПП.

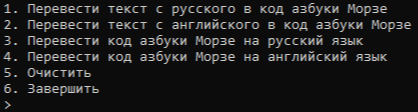


Рисунок 28. Меню

Пункты перевода работают однотипно. Каждый пункт предлагает загрузить файл, который пользователь хочет перевести. В случае выбора загрузки файла, программа запросит его название. Адрес файла может вводиться как относительно расположения исполняемого файла, так и напрямую, но он должен содержать не более 255 символов вместе с названием файла. Если программа не смогла открыть файл, то она выводит соответствующее сообщение об ошибке. В случае успеха программа переходит в режим перевода и выводит полученный текст.

Если пользователь хочет ввести вручную текст, то следует отказаться от загрузки файла. Тогда программа запросит текст для перевода. Если пользователь ничего не введёт, то программа пропустит перевод и вызовет меню.

После перевода программа предложит сохранить файл. Если пользователь хочет сохранить файл, то ПП запросит название файла. Если файл не существовал, то программа создаст его сама. Если файл существует, то программа запишет перевод в конец.

Также в меню есть пункт «очистка», которая очищает консоль. После исполнения этого пункта вновь вызывается меню.

Требования к файлам для перевода с русского языка:

1. Может содержать буквы кириллицы (без «ё») в верхнем и нижнем регистре
2. Может содержать цифры
3. Может содержать знаки «.», «,», «!», «?», «’», «”», «;», «:», «-», «+», «=», «\_», «/», «(», «)», «&», «$», «@».

Требования к файлам для перевода с английского языка:

1. Может содержать буквы латиницы в верхнем и нижнем регистре
2. Может содержать цифры
3. Может содержать знаки «.», «,», «!», «?», «’», «”», «;», «:», «-», «+», «=», «\_», «/», «(», «)», «&», «$», «@».

Требования к файлам для перевода с азбуки Морзе:

1. Код должен быть представлен с использованием «-», «\*» и пробела.
2. Расстояние между комбинациями 1 пробел
3. Расстояние между словами, представленными в Морзе – 2 пробела. (Если пробелов будет больше, то программа переведёт, но между символами будет больше пробелов)

Программа распознаёт табуляцию и перевод строки.

# **Заключение**

В ходе курсового проекта был получен ПП, способный переводить тексты с Морзе на русский/английский и обратно. Программа корректно обрабатывает все рассмотренные ситуации, которые могут возникнуть. Также выполняется корректный перевод.

В перспективе развития можно сделать более удобный интерфейс, оптимизировать программу, добавить новые языки для перевода и многое другое.

Программа может использоваться не только для перевода, но и для проверки правильности ручного перевода. ПП может также принимать участие в учебном процессе. Например, результаты тестирования студентов можно сравнить с переводом программы, что облегчит проверку, или пропустить их перевод обратно и сверить уже исходный текст с переводом студентов пропущенным через программу.

# **Литература**

1. <https://ru.wikipedia.org/wiki/Азбука_Морзе> - Азбука Морзе, материал из википедии.
2. <https://calcsbox.com/post/perevodcik-azbuki-morze-onlajn.html> - Переводчик азбуки Морзе.
3. <http://hostciti.net/calc/it/translation-system.html> - Перевод из одной системы счисления в другую
4. <https://translate.yandex.ru/> - Яндекс переводчик
5. <https://gsgen.ru/tools/fish-text/> - генератор текста на русском
6. <https://coolgenerator.com/random-text-generator> - генератор текста на английском

# **Приложение А**

1. #include<stdlib.h> // для работы с памятью
2. #include<stdio.h> // стандартная библиотека си
3. #include<string.h> // для работы со строками
4. #define size 96 // размер массива структур
5. #define N 6 //количество пунктво меню
6. struct letter{
7. char RuLetter;
8. char EnLetter;
9. char Morse[10];
10. };
11. struct letter translate[] = {{'А', 'A', "\*-"}, {'Б', 'B', "-\*\*\*"}, {'В', 'W', "\*--"},
12. {'Г', 'G', "--\*"}, {'Д', 'D', "-\*\*"}, {'Е', 'E', "\*"},
13. {'Ж', 'V', "\*\*\*-"}, {'З', 'Z', "--\*\*"}, {'И', 'I', "\*\*"},
14. {'Й', 'J', "\*---"}, {'К', 'K', "-\*-"}, {'Л', 'L', "\*-\*\*"},
15. {'М', 'M', "--"}, {'Н', 'N', "-\*"}, {'О', 'O', "---"},
16. {'П', 'P', "\*--\*"}, {'Р', 'R', "\*-\*"}, {'С', 'S', "\*\*\*"},
17. {'Т', 'T', "-"}, {'У', 'U', "\*\*-"}, {'Ф', 'F', "\*\*-\*"},
18. {'Х', 'H', "\*\*\*\*"}, {'Ц', 'C', "-\*-\*"}, {'Ч', '#', "---\*"},
19. {'Ш', '#', "----"}, {'Щ', 'Q', "--\*-"}, {'Ъ', '#', "--\*--"},
20. {'Ы', 'Y', "-\*--"}, {'Ь', 'X', "-\*\*-"}, {'Э', '#', "\*\*-\*\*"},
21. {'Ю', '\*', "\*\*--"}, {'Я', '#', "\*-\*-"}, {')', ')', "-\*--\*-"},
22. {'а', 'a', "\*-"}, {'б', 'b', "-\*\*\*"}, {'в', 'w', "\*--"},
23. {'г', 'g', "--\*"}, {'д', 'd', "-\*\*"}, {'е', 'e', "\*"},
24. {'ж', 'v', "\*\*\*-"}, {'з', 'z', "--\*\*"}, {'и', 'i', "\*\*"},
25. {'й', 'j', "\*---"}, {'к', 'k', "-\*-"}, {'л', 'l', "\*-\*\*"},
26. {'м', 'm', "--"}, {'н', 'n', "-\*"}, {'о', 'o', "---"},
27. {'п', 'p', "\*--\*"}, {'р', 'r', "\*-\*"}, {'с', 's', "\*\*\*"},
28. {'т', 't', "-"}, {'у', 'u', "\*\*-"}, {'ф', 'f', "\*\*-\*"},
29. {'х', 'h', "\*\*\*\*"}, {'ц', 'c', "-\*-\*"}, {'ч', '#', "---\*"},
30. {'ш', '#', "----"}, {'щ', 'q', "--\*-"}, {'ъ', '#', "--\*--"},
31. {'ы', 'y', "-\*--"}, {'ь', 'x', "-\*\*-"}, {'э', '#', "\*\*-\*\*"},
32. {'ю', '\*', "\*\*--"}, {'я', '#', "\*-\*-"}, {'1', '1', "\*----"},
33. {'2', '2', "\*\*---"}, {'3', '3', "\*\*\*--"}, {'4', '4', "\*\*\*\*-"},
34. {'5', '5', "\*\*\*\*\*"}, {'6', '6', "-\*\*\*\*"}, {'7', '7', "--\*\*\*"},
35. {'8', '8', "---\*\*"}, {'9', '9', "----\*"}, {'0', '0', "-----"},
36. {'.', '#', "\*\*\*\*\*\*"}, {',', '.', "\*-\*-\*-"}, {'!', ',', "--\*\*--"},
37. {'#', '!', "-\*-\*--"}, {'?', '?', "\*\*--\*\*"}, {'\'', '\'', "\*----\*"},
38. {'"', '"', "\*-\*\*-\*"}, {';', ';', "-\*-\*-\*"}, {':', ':', "---\*\*\*"},
39. {'-', '-', "-\*\*\*\*-"}, {'+', '+', "\*-\*-\*"}, {'\_', '\_', "\*\*--\*-"},
40. {'/', '/', "-\*\*-\*"}, {'#', '(', "-\*--\*"}, {'(', ')', "-\*--\*-"},
41. {'&', '&', "\*-\*\*\*"}, {'$', '$', "\*\*\*-\*\*-"}, {'@', '@', "\*--\*-\*"},
42. {' ', ' ', ' '}, {'\n', '\n', '\n'}, {'\t', '\t', '\t'} };
43. const char \* const Menu[] = {"none",
44. "Перевести текст с русского в код азбуки Морзе",
45. "Перевести текст с английского в код азбуки Морзе",
46. "Перевести код азбуки Морзе на русский язык",
47. "Перевести код азбуки Морзе на английский язык",
48. "Очистить",
49. "Завершить"};
50. int DoMenu(char \* s[], int max); //Функция вызова меню
51. void EnteringString(char \*\*t, int \*len); // Функция ввода строки
52. void Print(char \*text); // Функция вывода введённого текста
53. void PrintMorze(struct letter Morze[], int len); //функция вывода морзе
54. void PrintRu(struct letter \*Morze, int len); // функция вывода русского языка
55. void PrintEn(struct letter \*Morze, int len); // функция вывода английского языка
56. void RussianToMorseTranslation(char \*text, int len, struct letter \* Morze); // перевод с русского в морзе
57. void EnglishToMorseTranslation(char \*text, int len, struct letter \* Morze); // перевод с английского в морзе
58. int CountAndSplitLettersMorse(char \*text, int len, struct letter \*\*Morze); // подсчёт и разделение на буквы
59. void MorseToRussianTranslation(struct letter \*Morze, int count); // перевод с морзе на русский
60. void MorseToEnglishTranslation(struct letter \*Morze, int count); // перевод с морзе на английский
61. void SaveFileMorze(struct letter \*Morze, int len); // сохранение в файл морзе
62. void SaveFileRu(struct letter \*Morze, int len); // сохранение в файл русский
63. void SaveFileEn(struct letter \*Morze, int len); // сохранение в файл английский
64. void LoadFile(char \*\*t, int \*len); // загрузка файла
65. int main(){
67. system("chcp 1251>NUL");
69. int k;
70. char \*text;
71. int len = 1;
72. int count = 0;
73. char choice;
75. struct letter \*Morze;
77. do{
78. k = DoMenu((char \*\*)Menu, N);
79. switch(k){
80. case 1: // Перевести текст с русского в код азбуки Морзе
81. printf("Загрузить файл?\n(y/n) ");
82. scanf("%c", &choice);
83. fflush(stdin);
84. if (choice == 'y')
85. LoadFile(&text, &len);
86. else
87. EnteringString(&text, &len);
88. if(\*text){
89. Morze = (struct letter \*)malloc(len \* sizeof(struct letter));
90. RussianToMorseTranslation(text, len, Morze);
91. PrintMorze(Morze, len-1);
92. printf("Сохранить?\n(y/n) ");
93. scanf("%c", &choice);
94. if (choice == 'y')
95. SaveFileMorze(Morze, len-1);
96. }
97. break;
98. case 2: // Перевести текст с английского в код азбуки Морзе
99. printf("Загрузить файл?\n(y/n) ");
100. scanf("%c", &choice);
101. fflush(stdin);
102. if (choice == 'y')
103. LoadFile(&text, &len);
104. else
105. EnteringString(&text, &len);
106. if (\*text){
107. Morze = (struct letter \*)malloc(len \* sizeof(struct letter));
108. EnglishToMorseTranslation(text, len, Morze);
109. PrintMorze(Morze, len-1);
110. printf("Сохранить?\n(y/n) ");
111. scanf("%c", &choice);
112. if (choice == 'y')
113. SaveFileMorze(Morze, len-1);
114. }
115. break;
116. case 3: // Перевести код азбуки Морзе на русский язык"
117. printf("Загрузить файл?\n(y/n) ");
118. scanf("%c", &choice);
119. fflush(stdin);
120. if (choice == 'y')
121. LoadFile(&text, &len);
122. else
123. EnteringString(&text, &len);
124. if (\*text){
125. Morze = (struct letter \*)malloc(sizeof(struct letter));
126. count = CountAndSplitLettersMorse(text, len, &Morze);
127. MorseToRussianTranslation(Morze, count);
128. PrintRu(Morze, count+1);
129. printf("Сохранить?\n(y/n) ");
130. scanf("%c", &choice);
131. if (choice == 'y')
132. SaveFileRu(Morze, count);
133. }
134. break;
135. case 4: // Перевести код азбуки Морзе на английский язык
136. printf("Загрузить файл?\n(y/n) ");
137. scanf("%c", &choice);
138. fflush(stdin);
139. if (choice == 'y')
140. LoadFile(&text, &len);
141. else
142. EnteringString(&text, &len);
143. if (\*text){
144. Morze = (struct letter \*)malloc(sizeof(struct letter));
145. count = CountAndSplitLettersMorse(text, len, &Morze);
146. MorseToEnglishTranslation(Morze, count);
147. PrintEn(Morze, count+1);
148. printf("Сохранить?\n(y/n) ");
149. scanf("%c", &choice);
150. if (choice == 'y')
151. SaveFileEn(Morze, count);
152. }
153. break;
154. case 5: // Очистить
155. system("cls");
156. break;
157. }
158. }while(k != N);
160. free(text);
161. free(Morze);
163. system("pause");
164. return 0;
165. }
166. int DoMenu(char \* s[], int max){
167. int i = 1;
168. for (i = 1; i <= max; i++)
169. printf(" %d. %s\n", i , s[i]);
170. printf(" >");
171. fflush(stdin);
172. scanf("%d", &i);
173. fflush(stdin);
174. return i<1||i>max?0:i;
175. }
176. void EnteringString(char \*\*t, int \*len){
177. char c;
178. \*len = 1;
179. \*t = (char \*)malloc(sizeof(char));
180. printf("Введите ваш текст: ");
181. while((c = getchar()) != '\n' && \*t != NULL){
182. (\*t)[(\*len) - 1] = c;
183. (\*len)++;
184. \*t = (char \*)realloc((\*t), (\*len));
185. }
186. if(\*t != NULL)
187. (\*t)[(\*len)-1] = '\0';
188. }
189. void Print(char \*text){
190. printf("Ваш текст:\n%s\n", text);
191. }
192. void PrintMorze(struct letter \*Morze, int len){
193. printf("Перевод в морзе:\n");
194. for(int k = 0; k < len; k++){
195. printf("%s ", Morze[k].Morse);
196. }
197. printf("\n");
198. }
199. void PrintRu(struct letter \*Morze, int len){
200. printf("Перевод на русский:\n");
201. for(int k = 0; k < len-1; k++){
202. printf("%c", Morze[k].RuLetter);
203. }
204. printf("\n");
205. }
206. void PrintEn(struct letter \*Morze, int len){
207. printf("Перевод на английский:\n");
208. for(int k = 0; k < len-1; k++){
209. printf("%c", Morze[k].EnLetter);
210. }
211. printf("\n");
212. }
213. void RussianToMorseTranslation(char \*text, int len, struct letter \* Morze){
214. for(int i = 0; i < len; i++){
215. for(int j = 0; j < size; j++){
216. if(text[i] == translate[j].RuLetter){
217. Morze[i].RuLetter = text[i];
218. strcpy(Morze[i].Morse, translate[j].Morse);
219. break;
220. }
221. else{
222. Morze[i].RuLetter = text[i];
223. strcpy(Morze[i].Morse, "#");
224. }
225. }
226. }
227. }
228. void EnglishToMorseTranslation(char \*text, int len, struct letter \* Morze){
229. for(int i = 0; i < len; i++){
230. for(int j = 0; j < size; j++){
231. if(text[i] == translate[j].EnLetter){
232. Morze[i].EnLetter = text[i];
233. strcpy(Morze[i].Morse, translate[j].Morse);
234. break;
235. }
236. else{
237. Morze[i].EnLetter = text[i];
238. strcpy(Morze[i].Morse, "#");
239. }
240. }
241. }
242. }
243. int CountAndSplitLettersMorse(char \*text, int len, struct letter \*\*Morze){
245. int count = 0;
246. int lenletter = 0;
247. for(int i = 0; i < len-1; i++){
248. if(text[i] == '\n' || text[i] == '\t' || text[i] == ' '){
249. (\*Morze)[count].Morse[lenletter] = '\0';
250. count++;
251. lenletter = 0;
252. \*Morze = (struct letter \*)realloc(\*Morze, (count+1)\*sizeof(struct letter));
253. if(text[i] == '\n' || text[i] == '\t'){
254. (\*Morze)[count].Morse[lenletter] = text[i];
255. count++;
256. \*Morze = (struct letter \*)realloc(\*Morze, (count+1)\*sizeof(struct letter));
257. }
258. continue;
259. }
260. if(lenletter < 10){
261. (\*Morze)[count].Morse[lenletter] = text[i];
262. lenletter++;
263. }
264. }
265. (\*Morze)[count].Morse[lenletter] = '\0';
266. return count+1;
267. }
268. void MorseToRussianTranslation(struct letter \*Morze, int count){
269. for(int i = 0; i <= count; i++){
270. for(int j = 0; j < size; j++){
271. if(strcmp(Morze[i].Morse, translate[j].Morse) == 0){
272. Morze[i].RuLetter = translate[j].RuLetter;
273. break;
274. }
275. else if(Morze[i].Morse[0] == '\0'){
276. Morze[i].RuLetter = ' ';
277. break;
278. }
279. else{
280. Morze[i].RuLetter = '#';
281. }
282. }
283. }
284. }
285. void MorseToEnglishTranslation(struct letter \*Morze, int count){
286. for(int i = 0; i <= count; i++){
287. for(int j = 0; j < size; j++){
288. if(strcmp(Morze[i].Morse, translate[j].Morse) == 0){
289. Morze[i].EnLetter = translate[j].EnLetter;
290. break;
291. }
292. else if(Morze[i].Morse[0] == '\0'){
293. Morze[i].EnLetter = ' ';
294. break;
295. }
296. else{
297. Morze[i].EnLetter = '#';
298. }
299. }
300. }
301. }
302. void SaveFileMorze(struct letter \*Morze, int len){
304. FILE \*fp;
305. char file\_name[255];
306. printf("Введите имя файла: ");
307. scanf("%s", file\_name);
308. getchar();
309. printf("Имя файла: %s\n", file\_name);
311. if(fp = fopen(file\_name, "a")){
312. printf("Файл открыт\n");
313. for(int i = 0; i < len; i++){
314. fprintf(fp, "%s ", Morze[i].Morse);
315. }
316. fprintf(fp, "\n");
317. printf("Файл успешно сохранён\n");
318. }else{
319. printf("Не удалось открыть файл.\n");
320. }
322. fclose(fp);
323. }
324. void SaveFileRu(struct letter \*Morze, int len){
326. FILE \*fp;
327. char file\_name[255];
328. printf("Введите имя файла: ");
329. scanf("%s", file\_name);
330. getchar();
331. printf("Имя файла: %s\n", file\_name);
333. if(fp = fopen(file\_name, "a")){
334. printf("Файл открыт\n");
335. for(int i = 0; i < len; i++){
336. fprintf(fp, "%c", Morze[i].RuLetter);
337. }
338. fprintf(fp, "\n");
339. printf("Файл успешно сохранён\n");
340. }else{
341. printf("Не удалось открыть файл.\n");
342. }
344. fclose(fp);
345. }
346. void SaveFileEn(struct letter \*Morze, int len){
348. FILE \*fp;
349. char file\_name[255];
350. printf("Введите имя файла: ");
351. scanf("%s", file\_name);
352. getchar();
353. printf("Имя файла: %s\n", file\_name);
355. if(fp = fopen(file\_name, "a")){
356. printf("Файл открыт\n");
357. for(int i = 0; i < len; i++){
358. fprintf(fp, "%c", Morze[i].EnLetter);
359. }
360. fprintf(fp, "\n");
361. printf("Файл успешно сохранён\n");
362. }else{
363. printf("Не удалось открыть файл.\n");
364. }
366. fclose(fp);
367. }
368. void LoadFile(char \*\*t, int \*len){
369. char c;
370. \*len = 1;
371. \*t = (char \*)malloc(sizeof(char));
373. FILE \*fp;
374. char file\_name[255];
375. printf("Введите имя файла: ");
376. scanf("%s", file\_name);
377. fflush(stdin);
378. // printf("Имя файла: %s\n", file\_name);
380. if(fp = fopen(file\_name, "r")){
381. while((c = fgetc(fp)) != EOF && \*t != NULL){
382. (\*t)[(\*len) - 1] = c;
383. (\*len)++;
384. \*t = (char \*)realloc((\*t), (\*len));
385. }
386. if(\*t != NULL)
387. (\*t)[(\*len)-1] = '\0';
388. }else{
389. printf("Не удалось открыть файл.\n");
390. }
392. fclose(fp);
393. }