

**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 Структурное программирование**

**Выполнил: студент группы 609-31
Кирьянов Д.И.**

**Проверил: доцент кафедры
Автоматики и
компьютерных систем
Гришмановский П.В.**

**Сургут
2024**

Задание.

Программа-переводчик.

Программа должна представлять собой переводчик текста с выбранного разработчиком языка на тот, который также выберет разработчик (в данном случае с русского на английский). Должна сохраняться пунктуация, а нераспознанные слова и нераспознанные последовательности остаться без изменений. Также необходимо предусмотреть возможность дополнять базу слов средствами самой программы или внешними средствами.

Изложение.

Тема курсового проекта: Программа-переводчик.

Выполнил: Кирьянов Д. И.

В данном курсовом проекте представлена разработка программного обеспечения «Программа-переводчик», предназначенного для перевода текста с русского языка на английский. Представленная «Программа-переводчик» наряду с переводом текста с русского языка на английский способна применять при своем функционировании и новые слова, ранее ей неизвестные, которые пользователь может загрузить в процессе ее использования. Кроме того, программа обладает функцией редактирования существующих записей, сохранения и загрузку пользовательского словаря.

Во введении раскрываются цели и задачи проекта, подчеркивается актуальность и значимость разработки «Программы-переводчика» в условиях глобализации и увеличения потребности в быстром и точном переводе текста.

В разделе исследования проведен сравнительный анализ функционирования и удобства при использовании существующих в настоящее время переводчиков.

Раздел разработки посвящен разработке архитектуры программы, описанию основных модулей и алгоритмов, на основе которых была реализована «Программа-переводчик». В этом разделе также описаны структуры данных и взаимодействие между компонентами системы.

В разделе написания кода представлены функции кода, написанные на языке С.

Раздел тестирования включает описание выбранных методов, способов и стратегий тестирования, разработанные тестовые сценарии, результаты проведенного тестирования и выявленные ошибки.

В разделе руководства для пользователей разработаны методические рекомендации и инструкции по установке, настройке и использованию программы, в том числе указаны проблемы, с которыми может столкнуться пользователь и пути их решения.

В заключении обобщены выводы по достоинствам и недостаткам в разработке программного обеспечения «Программа-переводчик».

Содержание.

Введение	5
Исследование.....	6
Разработка.....	9
Написание кода	15
Тестирование	24
Руководство для пользователя	26
Заключение	29
Список используемых источников.....	31
Листинг	32

Введение.

В современном мире, в условиях развития информационных технологий у людей различных национальностей появилась возможность коммуницировать, находясь в тысячах километрах друг друга; изучать информацию, находящуюся в просторах сети «Интернет», развиваться в различных направлениях. Однако, на первоначальном этапе развития информационных технологий встала проблема барьера в познании людей ввиду отсутствия качественного перевода, не позволяющего понять смысл содержания текста. Эта проблема была решена появлением незаменимых помощников в этом вопросе – переводчиков, способных быстро и точно преодолевать языковые барьеры, облегчая обмен информацией и знаниями между людьми, говорящими на разных языках.

В рамках проекта поставлена задача создания «Программы-переводчика», позволяющей осуществлять перевод текста, написанного на русском языке, на английский. Перед программным обеспечением стоит важная задача - точный и быстрый перевод с сохранением при этом смысла исходного текста.

«Программа-переводчик» — это простая программа на языке C, которая позволяет пользователям переводить отдельные слова и тексты, не прибегая к сложным инструментам и профессиональным переводчикам.

Целью проекта «Программа-переводчик» является разработка функционального, но в тоже время и простого, точного и быстрого переводчика, способного обрабатывать текстовые данные и предоставлять качественный перевод.

В ходе курсового проекта будут решены следующие задачи:

1. Ознакомление с особенностями предметной области перевода текстов.
2. Изучение и разбор программ-аналогов, выявление их достоинств и недостатков.
3. Создание алгоритмов перевода и структур данных для хранения словарей.
4. Реализация программного обеспечения.
5. Тестирование программы для обеспечения ее корректной работы.
6. Анализ полученных результатов, дальнейшая его корректировка.
7. Подготовка технической документации на разработанное программное обеспечение.

Таким образом, ожидаемым результатом курсового проекта является создание качественного, быстрого, простого в использовании программного обеспечения «Программа-переводчик», способного эффективно переводить текст с русского на английский язык. Данная разработка в дальнейшем облегчит осуществления чтения информации на непонятном для пользователя языке при осуществлении образовательного процесса, выполнения задач, возложенных на работника, а также в повседневной деятельности людей.

Исследование.

Перед началом работы над «Программой-переводчиком», нужно тщательно изучить данную тему, уточнить существующие программы-аналоги, а также установить требования к нашей программе.

Тема данного курсового проекта охватывает перевод текстов с русского языка на английский, и для успешной реализации программы необходимо уточнение следующих концепций:

1. Лексические, синтаксические, пунктуационные особенности русского и английского языков.
2. Принципы машинного перевода и алгоритмы обработки естественного языка.
3. Способы обработки и хранения словарей, а также методы добавления и редактирования новых слов и выражений.

Для понимания текущего положения в области перевода необходимо проанализировать существующие решения, в первую очередь будет видно их достоинства и недостатки, а также поймем, какие использовать идеи в разработке.

1. Google Translate

Google Translate – это онлайн-сервис, который предлагает перевод между большим количеством языков и включает в себя не только текстовый перевод, но и голосовой, а также перевод изображений. Интерфейс интересен и прост, а результаты перевода предоставляются моментально. Несмотря на хорошие достоинства данного онлайн-сервиса, он имеет и существенный недостаток – зависимость работы от подключения к Интернету, что может стать проблемой для пользователей с ограниченным или нестабильным доступом к сети. Также google translate имеет свой API для интеграции в разного рода проекта, так, например, был личный интерес интеграции сервиса google в очки для предоставления информации в реальном времени. Но что использование API, что интеграция в проекты не нужны, поскольку доступа в интернет может и не быть.

2. Яндекс Переводчик

Яндекс Переводчик схож по своему содержанию и функционалу с Google Translate. Он аналогично удобен и просто в использовании, осуществляет перевод не только текста, но

и голосовых текстом и изображений. Также имеется проблема зависимости от подключения к сети «Интернет», ограничивающий его использование неопределенным кругом лиц.

3. Microsoft Translator

Microsoft Translator – это приложение, доступное на различных платформах, осуществляющее перевод текста, голоса и изображений. Главное его отличие от вышеупомянутых онлайн-сервисов – это интеграция с другими продуктами Microsoft. Немаловажным достоинством является возможность пользователю работать в офлайн-режиме, загружая необходимые языковые пакеты, но и здесь не без подводных камней - для использования на компьютере без установки дополнительных программ может потребоваться интернет-соединение для первоначальной настройки и загрузки необходимых данных. Еще одним недостатком данного приложения является неудачно подобранный интерфейс, который не нравится многим пользователям, вследствие чего малое количество людей предпочитают пользоваться именно Microsoft Translator, а не его более удачными конкурентами - google translate и Яндекс переводчиком.

Анализируя эти программы-аналоги, становится ясно, что необходимо создать программу-переводчик, которая будет включать все достоинства вышеупомянутых сервисов и избегать дублирование недостатков. Таким образом, она должна быть удобна в использовании, способна выполнять точные переводы текстов с русского на английский и обладать возможностью работы в офлайн-режиме без подключения к сети при любых обстоятельствах. Более того, данный проект будет обладать возможностью подстраиваться под стиль и речь, которая будет удобна тому или иному пользователю. Исходя из этого разные люди могут создавать свои библиотеки для всеобщего пользования, что не может не радовать людей, которые не принимают один вариант перевода.

Таким образом, данный проект направлен на разработку функционального и удобного в использовании переводчика, который обеспечит высокое качество перевода, будет поддерживать добавление и редактирование словарей пользователями, делиться ими, а также сможет работать без подключения к сети «Интернет».

Для составления качественного алгоритма программы, имеется необходимость вынести то, что действительно необходимо реализовать в программе, а именно:

1. Последовательный ввод слов и фраз.
2. Перевод введенного текста с учетом контекста и пунктуации.
3. Результат предыдущего перевода может быть использован для следующего.
4. Простой и понятный интерфейс для любого пользователя.

Функционал программы:

1. Вывод полного списка слов и фраз в словаре.
2. Перевод текстов и слов.
3. Работа со словарем:
 - 3.1. Добавление нового слова.
 - 3.2. Добавление нового текста.
 - 3.3. Редактирование существующего слова.
 - 3.4. Удаление слова.
 - 3.5. Сохранение словаря в файл.
 - 3.6. Загрузка словаря из файла.
4. Обработка исключительных ситуаций.

Этап создания:

1. **Исследование предметной области и программ-аналогов.**
 - 1.2. Выявление ключевых требований к разрабатываемой программе.
2. **Разработка архитектуры программы.**
 - 2.2. Разработка алгоритмов перевода и обработки текста.
 - 2.3. Создание модулей для добавления, редактирования и удаления слов.
3. **Реализация программного обеспечения.**
4. **Тестирование и отладка.**
 - 4.2. Проведение тестирования и выявление ошибок.
5. **Подготовка технической документации.**
 - 5.1. Руководство пользователя по работе с программой.

Разработка.

Разработка – первоначальный этап деятельности по созданию программного обеспечения «Программа-переводчик», отвечающий за функционирование и сопровождение программных продуктов.

При включении данного программного обеспечения пользователю будет предложен список доступных функций и команд, таких как: перевод слова, перевод текста, добавление нового слова или текста, редактирование и удаление слов, а также сохранение и загрузка словаря в *текстовом формате*.

Каким образом будет работать программа. Пользователь вводит в отведенную для этого строку текст на русском языке для перевода. Программа разбивает текст на отдельные слова, обрабатывает каждый элемент, сохраняя при этом пунктуацию, пробелы и смысл всего содержания текста. Программа проверяет наличие каждого слова в ее словаре, если возникает ситуация, при которой какое-либо слово в словаре отсутствует, то пользователю предлагается ввести перевод для этого слова, который в дальнейшем сохраняется в словарь. Важно, что если при переводе слово в словаре отсутствует, то в этом случае текст будет переводить **ТОЛЬКО** те слова, которые имеются в словаре. В противном случае программа будет игнорировать слова, которые найдены не были, и пользователь сможет самостоятельно дополнить отсутствующие слова для дальнейшего использования.

Для реализации процесса перевода, программа будет использовать двусвязный список или массив структур, где каждая структура содержит слово на русском и его перевод на английский, это сделано для того, чтобы не было проблем после загрузки отдельных готовых словарей, поскольку слова могут либо теряться, либо путаться после перезапуска. Если во время перевода или добавления слова возникнет ошибка, пользователю будет выведено понятное сообщение об ошибке.

Программа будет работать до тех пор, пока пользователь не выберет команду выхода, либо не закроет самостоятельно свое окружение.

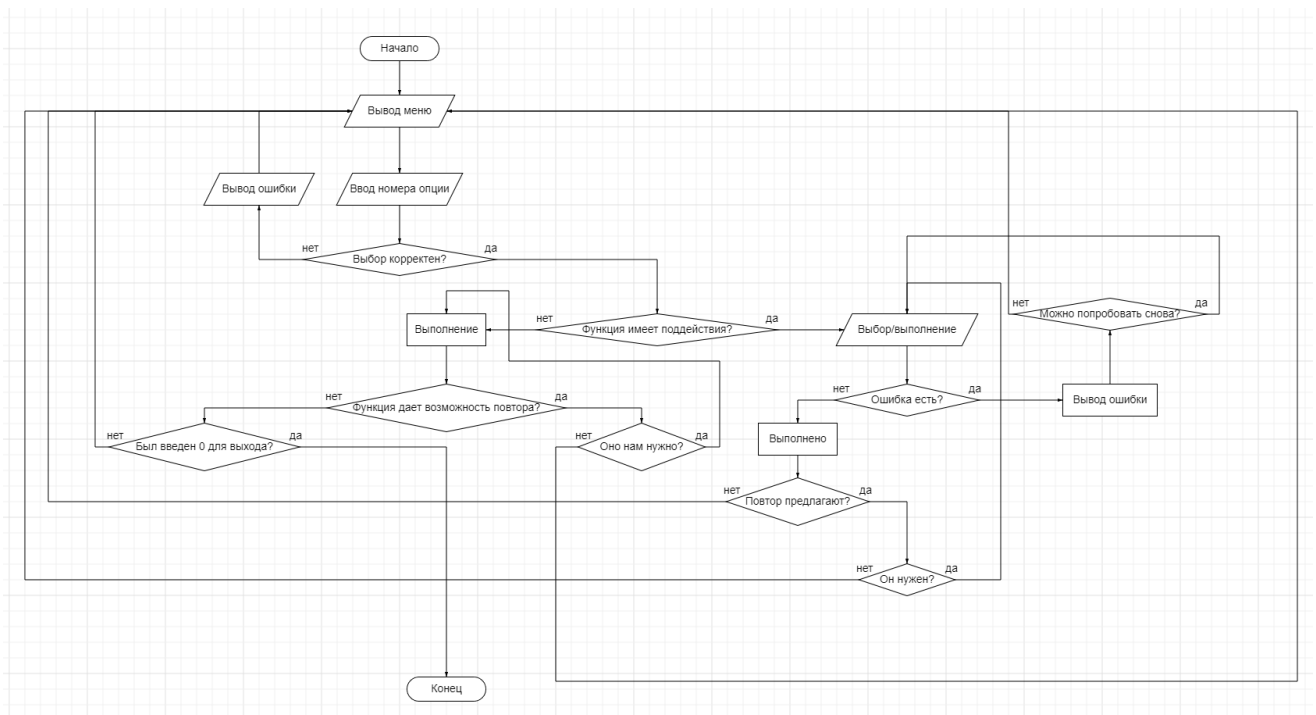


Рисунок 1. Блок-схема программы

Вот примерный алгоритм нашей программы:

Алгоритм:

1. Принять исходный текст.
2. Разделить текст на слова, не теряя синтаксические, лексические и пунктуационные особенности русского языка.
3. Для каждого слова:

Поиск слова в стандартном словаре.

Если слово найдено, добавить его перевод к результату.

Если слово не найдено, поиск в добавленном словаре.

Если слово не найдено, оставить слово без изменений.

4. Вернуть переведенный текст.

Добавление нового слова.

1. Принять слово на русском и его перевод на английском.
2. Привести первую букву каждого слова к верхнему регистру.
3. Проверить, если словарь добавленных слов заполнен, увеличить его объем.
4. Добавить слово и его перевод в словарь добавленных слов.
5. Уведомить сообщением пользователя об успешном добавлении слова в словарь.

Добавление нового текста

1. Принять входной текст.
2. Разделить текст на слова и сохранить.

3. Для каждого слова:

Поиск слова в стандартном словаре.

Если слово не найдено, поиск в добавленном словаре.

Если слово не найдено, запросить у пользователя перевод.

Добавить слово и его перевод в словарь добавленных слов.

4. Вернуть сообщение об успешном добавлении всех новых слов.

Редактирование существующего слова

1. Принять слово на русском для редактирования.

2. Поиск слова в стандартном словаре.

3. Если слово найдено, запросить новый перевод.

4. Обновить перевод в словаре.

5. Если слово не найдено в стандартном словаре, поиск в добавленном словаре.

6. Если слово найдено, обновить перевод.

7. Если слово не найдено, вернуть сообщение «Слово не найдено».

Удаление слова.

1. Принять слово на русском.

2. Поиск слова в стандартном словаре.

3. Если слово найдено, удалить его из словаря.

4. Если слово не найдено в стандартном словаре, поиск в добавленном словаре.

5. Если слово найдено, удалить его из словаря.

6. Если слово не найдено, вернуть сообщение «Слово не найдено».

Отображение словаря

1. Пройти по всем словам в стандартном словаре и вывести их на экран.

2. Пройти по всем словам в добавленном словаре и вывести их на экран.

Сохранение словаря

1. Принять имя файла.

2. Открыть файл для записи.

3. Записать заголовок «STANDARD_WORDS».

4. Записать все слова и переводы из стандартного словаря в файл.

5. Записать заголовок «ADDED_WORDS».

6. Записать все слова и переводы из добавленного словаря в файл.

7. Закрывать файл.

8. Уведомить сообщением пользователя об успешном сохранении.

Загрузка словаря.

1. Принять имя файла.

2. Открыть файл для чтения.
3. Освободить текущий словарь.
4. Инициализировать новый словарь.
5. Чтение из файла до заголовка «STANDARD_WORDS».
6. Считать слова и переводы в стандартный словарь до заголовка «ADDED_WORDS».
7. Считать слова и переводы в добавленный словарь.
8. Закрыть файл.
9. Уведомить сообщением пользователя об успешном загрузке.



Рисунок 2. Пример стандартной операции

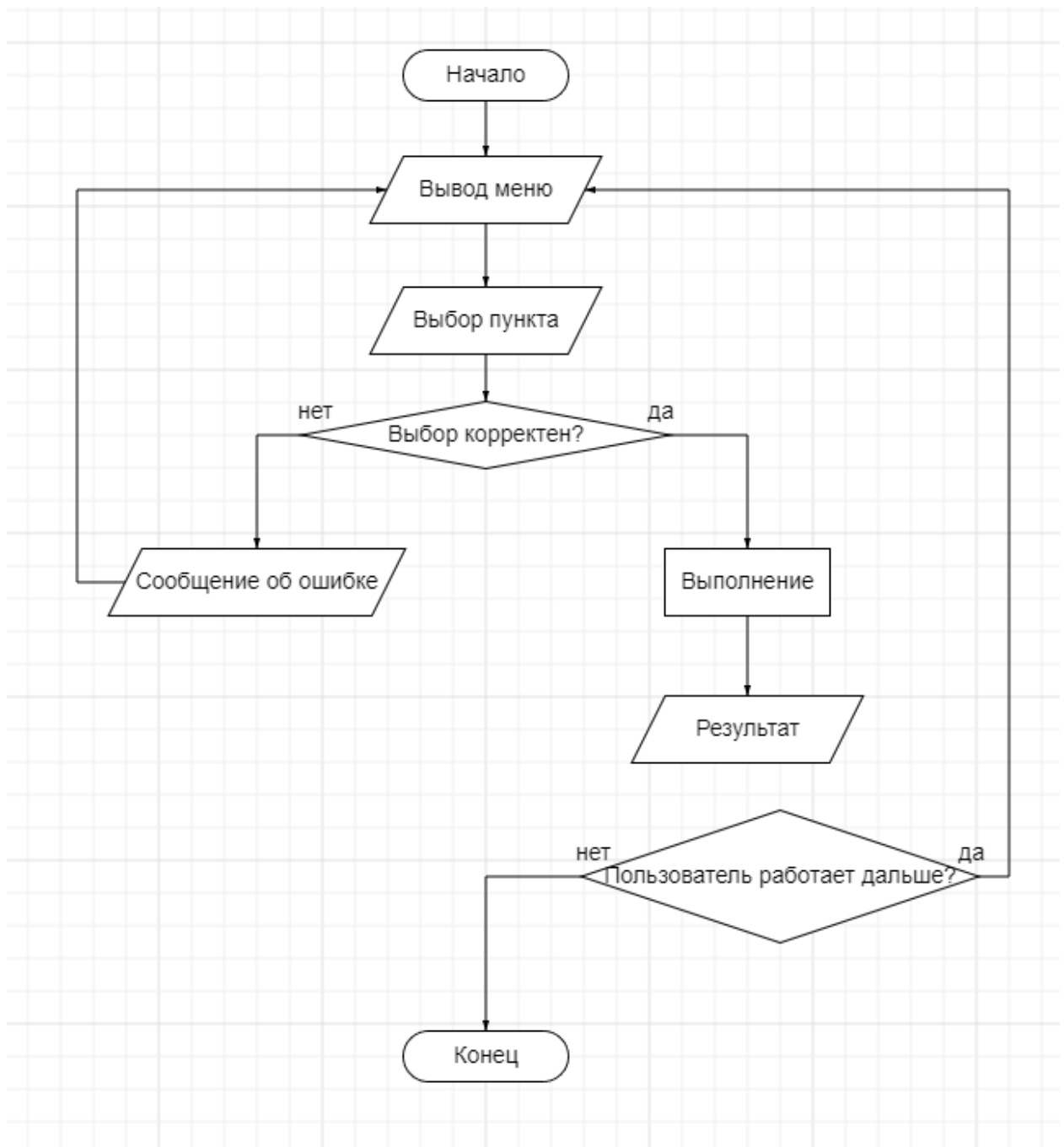


Рисунок 3. Принцип работы с операциями

Теперь готова структура, которой можно придерживаться для дальнейшей разработки, но также необходимо просмотреть аналоги, чтобы выбрать «золотую середину» для создания интерфейса.

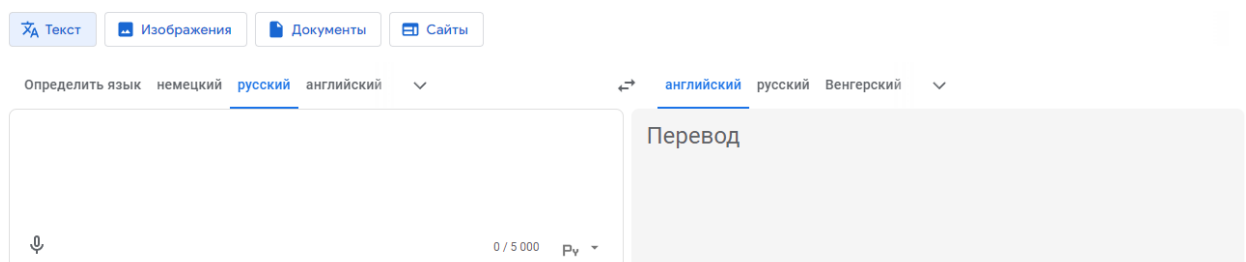


Рисунок 4. Интерфейс "Google translate"

Поскольку программа будет использоваться в пространстве терминала (далее будем говорить CMD), а также из-за отсутствия библиотек и возможности создания графического десктопного интерфейса, данный вариант можно назвать «Отличным», но не пригодным для нашей текущей задачи.

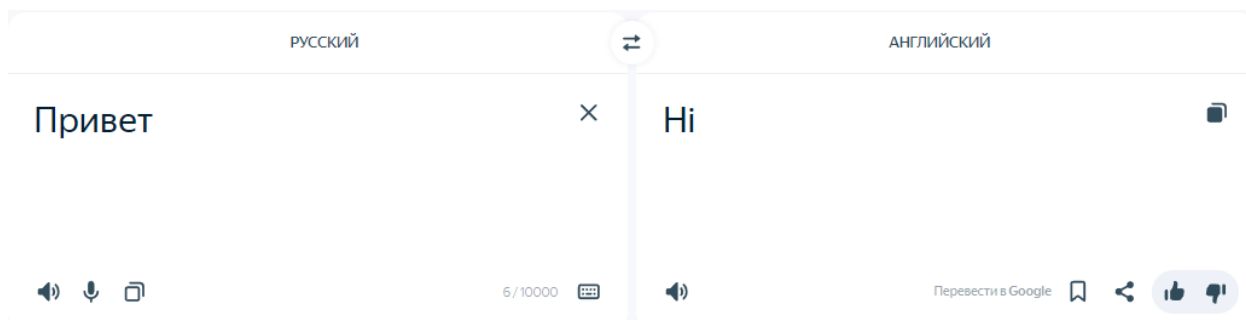


Рисунок 5. Интерфейс "Яндекс. Переводчик"

Также, как и с google translate, мало что можно взять из данного интерфейса, но единственное, что можно сделать похожее по интерфейсу – это наличие разбиения на пространство между текстом, который нужно перевести, и между готовым переводом. Про логику самого перевода комментарии излишни, функционал вышеперечисленных сервисов достигался очень большим количеством людей.

Совместив интерфейсы двух сервисов, выходит следующий результат:

```
Введите текст на русском: Привет, как ты?  
Переведенный текст: Hello, How You?
```

Рисунок 6. Готовый перевод предложений.

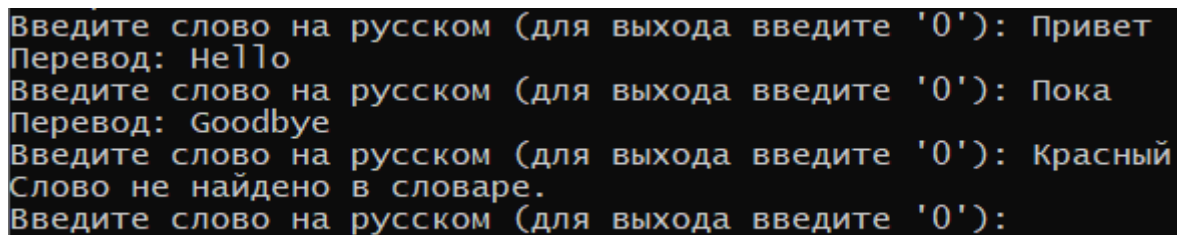
Поскольку программа написана для окружения CMD, по субъективному мнению, будет неудобно выводить всю информацию при переводе предложений в строку (исключение: перевод слов). Также можно заметить, что, к сожалению, отсутствуют такие возможности, как перевод по записи голоса, картинке или другими неизвестными мне методами. Программа поддерживает **ТОЛЬКО** текстовый формат.

```
Введите слово на русском (для выхода введите '0'): Привет  
Перевод: Hello  
Введите слово на русском (для выхода введите '0'): Пока  
Перевод: Goodbye  
Введите слово на русском (для выхода введите '0'): _
```

Рисунок 7. Пример перевода слова.

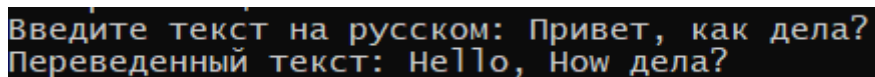
Поскольку слова не занимают большую площадь CMD, то в этом случае логичнее расположить перевод ниже для разделения, а также предоставить пользователю вновь проверять

слова, пока пользователю это станет не нужно, либо пользователь не попадет на слово, которое не знает словарь.



```
Введите слово на русском (для выхода введите '0'): Привет
Перевод: Hello
Введите слово на русском (для выхода введите '0'): Пока
Перевод: Goodbye
Введите слово на русском (для выхода введите '0'): Красный
Слово не найдено в словаре.
Введите слово на русском (для выхода введите '0'):
```

Рисунок 8. Попадание на неизвестное слово.



```
Введите текст на русском: Привет, как дела?
Переведенный текст: Hello, How дела?
```

В приведенном выше случае можно увидеть, что программа не знает слово «дела», и вместо того, чтобы выдавать ошибку или некорректное значение, в отдельных функциях предусмотрены несколько путей решений, которыми можно решить данный вопрос:

- 1) Записать отдельно слово/слова через добавление *Отдельного слова*.
- 2) Добавить полностью новый *Текст*, в котором пользователя попросят указать перевод всем неизвестным словам.

Если пользователь желает ознакомиться со всеми словами, которые когда-либо были введены, или пользователь захочет ознакомиться со словарем, то при выборе соответствующей опции сразу можно это сделать.

Написание кода.

Этап кодирования является самым важным, и самым ответственным моментов в данном проекте. В данном этапе идет создание функционала программы на основе исследований и идей проекта в программный код. Программа будет написана на языке C.

Выбор типа данных.

При разработке программы будем использовать текстовый тип данных для хранения слов и переводов. В языке C тип данных `wchar_t` является лучшим выбором для хранения, что необходимо для поддержки русского и английского языков. Использование массива структур, где каждая структура содержит слово на русском и его перевод на английский, поможет программе более гибко и правильно взаимодействовать со словарем.

Почему текстовый тип данных?

- **Гибкость:** позволяет легко управлять данными, такими как слова и предложения.
- **Читаемость:** упрощает исправления и проверку данных.
- **Память:** занимает немного меньше места.

Выбор символов Юникода.

Юникод - это международный стандарт кодирования символов, который взаимодействует с разными языками мира. Он позволяет использовать единую кодировку для всех символов, что очень выручает в тех моментах, когда дело доходит до работы с разными языковыми платформами.

Почему Юникод?

- **Многоязычная поддержка:** позволяет корректно отображать и обрабатывать символы разных языков без проблем.
- **Совместимость:** поддержание Юникода является достаточно повсеместным, что упрощает разработку и интеграцию в будущем, если она будет.

Взаимодействие с пользователем через CMD.

Для реализации программы потребуется взаимодействие с пользователем через CMD. Ввод и вывод данных будут осуществляться с использованием функции `stdio.h`.

Использование функций ввода и вывода

Для подключения функций ввода и вывода в программе необходимо включить заголовочный файл:

Использование функции ввода и вывода.

Для подключения функций ввода и вывода в программе необходимо подключить `stdio.h`:


```
#include <stdio.h>
```

stdio.h содержит определения и прототипы функций для стандартного ввода и вывода. Основные функции, которые будут использоваться, это `fgetws` и `wprintf`.

1. **Функция `fgetws`** используется для считывания строк в CMD. Она принимает указатель на массив `wchar_t` и размер массива.

Пример:

```
fgetws(input, 50, stdin);
```

2. **Функция `wprintf`** используется для вывода строк в CMD. Она принимает форматную строку и список аргументов.

Пример:

```
wprintf(L"Перевод: %ls\n", translation);
```

Работа с динамической памятью.

Программа будет активно использовать динамическое выделение памяти для хранения слов и переводов. Для этого будут использоваться функции `malloc`, `realloc` и `free` из `stdlib.h`.

Основные функции программы.

Для реализации программы-переводчика будут разработаны следующие функции:

1. Функция перевода слова:

Поиск слова в словаре.

Возврат перевода слова, если оно найдено, или сообщение о его отсутствии.

2. Функция перевода текста:

Разделение текста на слова с сохранением пунктуации.

Перевод каждого слова с использованием словаря.

Сохранение исходной пунктуации и пробелов в результирующем тексте.

3. Функция добавления нового слова:

Ввод слова на русском и его перевода на английский.

Приведение слов к нижнему регистру и капитализация первой буквы.

Добавление слова и его перевода в словарь.

4. Функция редактирования существующего слова:

Поиск слова в словаре.

Ввод нового перевода.

Обновление перевода в словаре.

5. Функция удаления слова:

Поиск слова в словаре.

Удаление слова и его перевода из словаря.

6. Функция отображения словаря:

Вывод всех слов и их переводов на экран.

7. **Функция сохранения словаря в файл:**

Ввод имени файла.

Запись слов и их переводов в файл.

8. **Функция загрузки словаря из файла:**

Ввод имени файла.

Чтение слов и их переводов из файла.

Освобождение текущего словаря и инициализация нового.

Примерный алгоритм основной функции.

Программа будет работать в цикле до тех пор, пока пользователь не выберет опцию выхода, либо пока не закроет CMD. Для этого будет использоваться бесконечный цикл `while (true)`, а обработка команд будет осуществляться с помощью оператора `switch`. В программе оператор `switch` используется для выбора функции, которая должна быть выполнена на основе ввода пользователя.

Синтаксис оператора switch.

```
switch () {  
    case 1:  
        // код  
        break;  
    case 2:  
        // код  
        break;  
    ...  
    default:  
        // код  
        break;  
}
```

Можем заметить, что синтаксис состоит из нескольких частей, а именно:

Ключ - Переменная, значение сравнение с `case`. Он может быть любого типа данных.

Значение – Постоянное значение, с которой сравнивается значение выражения.

Каждое значение `case` должно быть уникальным.

break - Оператор, используемый для выхода из оператора `switch` после выполнения соответствующего блока кода. Если `break` не используется, выполнение продолжится с первого оператора следующего `case`.

default - Блок кода, который выполняется по умолчанию, если значение выражения не совпадает ни с одним из значений `case`.

Основные компоненты программы.

1. Заголовочные файлы:

```
#include <stdio.h>
#include <wchar.h>
#include <locale.h>
#include <stdbool.h>
#include <wctype.h>
#include <stdlib.h>
#include <string.h>
```

Эти заголовочные файлы предоставляют функции для работы с широкими символами (Юникод), а также функции для ввода-вывода и управления памятью.

2. Функция main

Функция main является точкой входа в программу и управляет всеми основными операциями.

```
int main() {
    setlocale(LC_ALL, ".866");
    setlocale(LC_CTYPE, "ru_RU.UTF-8");
```

- **Установка кодировки:**

`setlocale(LC_ALL, ".866");` и `setlocale(LC_CTYPE, "ru_RU.UTF-8");` устанавливают кодировку для корректной работы с русскими символами. Также можно использовать `chsr` и указать необходимую кодировку.

```
Dictionary dict;
initializeDictionary(&dict);
```

- **Инициализация словаря:**

Создание переменной `dict` типа `Dictionary` и вызов функции `initializeDictionary` для стандартных и добавленных слов.

```
initializeStandardWords(&dict);
```

- **Заполнение словаря стандартными словами:**

Вызов функции `initializeStandardWords`, которая добавляет стандартные слова и их переводы в словарь.

```
int choice;
while (true) {
```

- **Начало бесконечного цикла:**

Объявляется переменная `choice`, которая будет использоваться для хранения выбора пользователя из меню.

`while (true)` запускает бесконечный цикл, который продолжается до тех пор, пока пользователь не выберет команду выхода, или не закроет CMD.

```
wprintf(L"Меню:\n");
wprintf(L"1. Узнать перевод с русского на английский\n");
wprintf(L"2. Перевести текст\n");
wprintf(L"3. Добавить новое слово\n");
wprintf(L"4. Добавить новый текст для перевода\n");
wprintf(L"5. Редактировать существующее слово\n");
wprintf(L"6. Удалить слово\n");
wprintf(L"7. Отобразить словарь\n");
wprintf(L"8. Сохранить словарь в файл\n");
wprintf(L"9. Загрузить словарь из файла\n");
wprintf(L"0. Выход\n");
wprintf(L"Выберите опцию: ");
wscanf(L"%d", &choice);
getchar(); // Удаление символа новой строки из входного
```

буфера

- **Вывод меню и считывание выбора пользователя:**

Выводится текст меню с доступными опциями.

`wscanf` считывает выбор пользователя и сохраняет его в переменной `choice`.

`getchar()` используется для удаления символа новой строки из входного буфера.

```
switch (choice) {
    case 1: {
        wchar_t input[50];
        while (true) {
            wprintf(L"Введите слово на русском (для выхода
введите '0'): ");
            fgetws(input, 50, stdin);
            wchar_t *newline = wcschr(input, L'\n');
            if (newline) *newline = L'\0';

            if (wcscmp(input, L"0") == 0) break;

            wchar_t translation[50];
            if (translateWord(input, &dict, translation))
            {
                wprintf(L"Перевод: %ls\n", translation);
            } else {
                wprintf(L"Слово не найдено в словаре.\n");
            }
        }
        break;
    }
}
```

Опция 1 - Узнать перевод с русского на английский:

Начало цикла для перевода слов.

Запрос ввода слова на русском.

Вызов `translateWord` для поиска перевода.

Вывод перевода, если слово найдено. В другом случае выводится сообщение о том, что слово не найдено.

Если введено '0', цикл прерывается, и возвращается в основное меню.

```
case 2: {
    wchar_t text[500];
    wprintf(L"Введите текст на русском: ");
    fgetws(text, 500, stdin);
    wchar_t *newline = wcschr(text, L'\n');
    if (newline) *newline = L'\0';

    translateText(text, &dict);
    break;
}
```

Опция 2 - Перевести слово:

Запрос ввода слова на русском.

Вызов `translateText` для перевода всего текста.

Возвращение в основное меню после перевода.

```
case 3:
    addWord(&dict);
    break;
```

Опция 3 - Добавить новое слово:

Вызов функции `addWord` для добавления нового слова в словарь.

Возвращение в основное меню после добавления.

```
case 4:
    addText(&dict);
    break;
```

Опция 4 - Добавить новый текст для перевода:

Вызов функции `addText` для добавления нового текста в словарь.

Возвращение в основное меню после добавления.

```
case 5:
    editWord(&dict);
    break;
```

Опция 5 - Редактировать существующее слово:

Вызов функции `editWord` для редактирования перевода существующего слова.

Возвращение в основное меню после редактирования.

```
case 6:
    deleteWord(&dict);
    break;
```

Опция 6 - Удалить слово:

Вызов функции `deleteWord` для удаления слова из словаря.

Возвращение в основное меню после удаления.

```
case 7:
    displayDictionary(&dict);
    break;
```

Опция 7 - Отобразить словарь:

Вызов функции `displayDictionary` для отображения всех слов и их переводов.

Возвращение в основное меню после отображения.

```
case 8:
    saveDictionary(&dict);
    break;
```

Опция 8 - Сохранить словарь в файл:

Вызов функции `saveDictionary` для сохранения текущего словаря в файл.

Возвращение в основное меню после сохранения.

```
case 9:
    loadDictionary(&dict);
    break;
```

Опция 9 - Загрузить словарь из файла:

Вызов функции `loadDictionary` для загрузки словаря из файла.

Возвращение в основное меню после загрузки.

```
case 0:
    freeDictionary(&dict);
    return 0;
```

Опция 0 - Выход:

Вызов функции `freeDictionary` для освобождения всей выделенной памяти.

Завершение программы.

```
default:
```

```

        wprintf(L"Некорректный выбор. Пожалуйста,
попробуйте снова.\n");
    }
}

```

Обработка некорректного выбора:

Если пользователь ввел некорректный номер опции, выводится сообщение об ошибке и цикл продолжается, предлагая снова сделать выбор.

```

        default:
            wprintf(L"Некорректный выбор. Пожалуйста,
попробуйте снова.\n");

```

Сообщение об ошибке: если введенный номер опции не соответствует ни одной из предопределенных опций, выводится сообщение об ошибке, и цикл продолжается.

Тестирование.

Тестирование программы поможет нам в доработке этапа разработки, направленным на систематическую проверку и оценку качества, функциональности, надежности и производительности. С помощью тестирования можно увидеть дефекты и отклонения.

Тестирование будет состоять из пунктов:

1. Тестирование по вводам-запросам.
2. Тестирование устойчивости программы на непредвиденные, или другие отклонения.

Тестирование на основе вводов.

Программа должна корректно переводить слова и тексты с русского на английский.

Тесты перевода отдельных слов.

Тест 1. «Привет», Ожидаемый результат: «Hello», Результат: «Hello» - Пройден.

Тест 2. Вход: «Спасибо», Ожидаемый результат: «Thank you», Результат: «Thank you» - Пройден.

Тест 3. «Утро», Ожидаемый результат: «Morning», Результат: «Morning» - Пройден.

Тест 4. «Извините», Ожидаемый результат: «Sorry», Результат: «Sorry» - Пройден.

Тест 5. «Да», Ожидаемый результат: «Yes», Результат: «Yes» - Пройден.

Тесты перевода текстов.

Тест 6. «Привет, как тебя зовут?», Ожидаемый результат: «Hello, what your name?», Результат: «Hello, what your name?» - Пройден.

Тест 7. «Спасибо, хорошего дня!», Ожидаемый результат: «Thank you, nice day!», Результат: «Thank you, nice day!» - Пройден.

Тест 8. «Доброе утро! Как спалось?», Ожидаемый результат: «Good morning! How sleep?», Результат: «Good morning! How sleep?» - Пройден.

Тест 9. «Извините за беспокойство», Ожидаемый результат: «Sorry for inconvenience», Результат: «Sorry for inconvenience» - Пройден.

Тест 10. «Да, конечно!», Ожидаемый результат: «Yes, of course!», Результат: «Yes, of course!» - Пройден.

Тесты обработок ошибок.

Тест 11. «слово1», Ожидаемый результат: «Ошибка: Слово не найдено в словаре.», Результат: «Слово не найдено в словаре.» - Пройден.

Тест 12. «слово123», Ожидаемый результат: «Слово не найдено в словаре», Результат: «Слово не найдено в словаре.» - Пройден.

Тест 13. «слово!», Ожидаемый результат: «Слово не найдено в словаре.», Результат: «Слово не найдено в словаре.» - Пройден.

Тест 14. «123», Ожидаемый результат: «Слово не найдено в словаре.», Результат: «Слово не найдено в словаре.» - Пройден.

Тест 15. «!@#», Ожидаемый результат: «Слово не найдено в словаре.», Результат: «Слово не найдено в словаре.» - Пройден.

Тестирование устойчивости

Тесты работ с некорректными значениями.

Тест 1. «х», Ожидаемый результат: «Слово не найдено в словаре.», Результат: «Ошибка: недопустимое слово» - Пройден.

Тест 2. «abc», Ожидаемый результат: «Слово не найдено в словаре.», Результат: «Ошибка: недопустимое слово» - Пройден.

Тест 3. «xnt», Ожидаемый результат: «Слово не найдено в словаре.», Результат: «Слово не найдено в словаре.» - Пройден.

Длительное использование и большое количество операций

Тест 4. Выполнить 50 операций, Ожидаемый результат: Программа должна продолжать работать без сбоев, Результат: Программа работала без сбоев - Пройден.

Тест 5. Выполнить 50 переводов текстов, Ожидаемый результат: Программа должна продолжать работать без сбоев, Результат: Программа работала без сбоев - Пройден.

Перевод длинного текста.

Тест 6. «Очень длинный текст для перевода», Ожидаемый результат: Переведенный текст должен соответствовать ожиданиям, Результат: Переведенный текст соответствует ожиданиям - Пройден. Пометка: Тест может быть успешно пройден, если все слова присутствуют или правильно записаны, в ином случае может возникнуть ошибка (некоторые слова не переведутся).

Тест 7. «Длинный текст с множеством символов и знаков препинания!!!», Ожидаемый результат: Переведенный текст должен соответствовать ожиданиям и сохранять исходные знаки препинания, Результат: Переведенный текст соответствует ожиданиям и сохраняет исходные знаки препинания - Пройден. Тест может быть успешно пройден, если все слова присутствуют или правильно записаны, в ином случае может возникнуть ошибка (некоторые слова не переведутся).

Руководство для пользователя.

Программа-переводчик — это программа, предназначенная для перевода слов и текстов с русского языка на английский. Программа поддерживает перевод отдельных слов и текстов, добавление новых слов и текстов, редактирование существующих переводов и сохранение/загрузку словаря. Результат предыдущего действия может использоваться как основа для следующего.

Минимальные системные требования

1. **Операционная система:** Windows 7 или выше. Возможна работа с дистрибутивами Linux.
2. **Процессор:** Intel или AMD с частотой 1 ГГц.
3. **Оперативная память:** 512 МБ.
4. **Свободное место на диске:** 10 МБ.
5. **Поддержка Unicode:** требуется для корректной обработки многоязычного текста.

Рекомендованные системные требования:

1. **Операционная система:** Windows 10. Возможна работа с дистрибутивами Linux.
2. **Процессор:** Intel Core i3 или AMD Ryzen 3 с частотой 2 ГГц.
3. **Оперативная память:** 2 ГБ.
4. **Свободное место на диске:** 50 МБ.
5. **Поддержка Unicode:** требуется для корректной обработки многоязычного текста.

Инструкция по установке:

1. Скачать исходный код и запустить его в любой среде.
2. Программа готова к работе.

Программа не требует дополнительных настроек.

Для начала работы с программой-переводчиком необходимо запустить программу и выбрать необходимую опцию в меню.

Правила эксплуатации с пояснениями:

1. Используйте программу только для тех целей, для которых она предназначена. Можете проверять программу на всевозможные варианты событий, но помните, что данная программа предназначена для простых действий. Использование программы для других целей может привести к непредсказуемым последствиям или повреждению данных.

2. Не вносите никаких изменений в исходный код программы.

Изменения в исходном коде могут привести к ошибкам, сбоям или к непредсказуемому действию. Создатель не несет ответственность за любые последствия, которые связаны по причине вмешательства в изначальный код. Предоставляется возможность проводить обновление исходного кода под личную ответственность пользователя, разрешается пользоваться программой, но если захотите разместить его в общедоступном пространстве, то, пожалуйста, свяжитесь для получения разрешения создателя (svortek@mail.ru, Кириянов Д.И.)

Сведения об обнаруженных недостатках программного продукта:

1. Программа может неверно переводить слова или тексты, если словарь не содержит соответствующего перевода. Программа-переводчик основана на статическом словаре, поэтому, если слово или выражение не найдено в словаре, пользователь должен добавить его вручную. Программа может не совсем адекватно реагировать на условия грамматики слов, где у одного слова может быть несколько приставочных слов (Пример: «Автобус» - «a bus». В данном примере самостоятельно пользователь указывает, нужно ли использовать приставки, в другом случае «a bus» будет записано как «Автобус».)

Предупреждения о возможном неадекватном функционировании:

1. Программа может вести себя неадекватно при работе с текстами, содержащими специфические символы или некорректные данные. Ввод некорректных данных или символов может привести к ошибкам в работе программы и неверным переводам. Если планируется устроить «Краш-тест» данной программе, пользователь должен быть готов к возможным последствиям в виде ошибок. В случае глобальных ошибок просьба связаться по вышеуказанным контактам.

Пример использования:

Основное меню:

1. Узнать перевод с русского на английский
2. Перевести текст
3. Добавить новое слово
4. Добавить новый текст для перевода
5. Редактировать существующее слово
6. Удалить слово
7. Отобразить словарь
8. Сохранить словарь в файл
9. Загрузить словарь из файла
0. Выход

Пример

1. Выберите опцию: 1 (Узнать перевод слова)
2. Введите слово на русском: «Привет»
3. Результат: «Hello»

Пример

1. Выберите опцию: 2 (Перевести текст)
2. Введите текст на русском: «Доброе утро!»
3. Результат: «Good morning!»

Важно

Для корректного функционирования программы-переводчика рекомендуется придерживаться данных инструкций и правил эксплуатации. Программа готова к использованию, но как было сказано выше, не стоит её целенаправленно ломать, для достижения наилучших результатов необходимо регулярно обновлять словарь и проверять корректность вводимых д.

Заключение.

Проект «Программа-переводчик» представляет собой функциональное и удобное программное обеспечение для перевода слов и текстов с русского на английский язык. В ходе разработки данного проекта были учтены все основные аспекты, необходимые для создания надежного и удобного инструмента, который может быть полезен в повседневной жизни, образовании и профессиональной деятельности.

Достоинства:

1. **Простота использования:** Интуитивно понятный интерфейс позволяет пользователям легко выполнять переводы слов и текстов. Основные функции программы доступны через простое меню в обычном CMD.
2. **Широкий спектр функциональности:** Программа поддерживает перевод отдельных слов, целых текстов, добавление и редактирование словарных записей. Возможность сохранения и загрузки словаря из файлов обеспечивает гибкость в использовании.
3. **Низкие системные требования:** Программа может работать на большинстве современных операционных систем с минимальными системными требованиями, что делает ее доступной для широкой аудитории.
4. **Руководство пользователя:** Подробное руководство пользователя помогает быстро освоить работу с программой, предоставляя инструкции по установке, настройке и использованию основных функций.

Недостатки:

1. **Ограниченная поддержка языков:** Программа ориентирована на перевод только с русского на английский язык..
2. **Ручное добавление новых слов:** Пользователю требуется вручную добавлять новые слова и их переводы в словарь.
3. **Отсутствие автоматического обновления словаря:** Программа не поддерживает автоматическое обновление словаря, что может привести к необходимости периодического ручного добавления новых слов и выражений.
4. **Зависимость от корректности пользовательского ввода:** Программа требует точного ввода данных пользователем. Некорректный ввод может привести к ошибкам в переводе.

5. **Ограниченная поддержка пунктуации и специальных символов:** Программа может не всегда корректно обрабатывать сложные случаи пунктуации и специальные символы, что может привести к ошибкам в переводе.

Программа протестирована, отредактирована, и предоставлена в функциональном виде, но дальнейшего потенциала для конкуренции с популярными сервисами, увы, не предоставляет.

Список используемых источников.

1. Википедия stdio.h - <https://ru.wikipedia.org/wiki/Stdio.h>
2. Википедия Указатели -
[https://ru.wikipedia.org/wiki/%D0%A3%D0%BA%D0%B0%D0%B7%D0%B0%D1%82%D0%B5%D0%BB%D1%8C_\(%D1%82%D0%B8%D0%BF_%D0%B4%D0%B0%D0%BD%D1%8B%D1%85\)](https://ru.wikipedia.org/wiki/%D0%A3%D0%BA%D0%B0%D0%B7%D0%B0%D1%82%D0%B5%D0%BB%D1%8C_(%D1%82%D0%B8%D0%BF_%D0%B4%D0%B0%D0%BD%D1%8B%D1%85))
3. Википедия Unicode -
<https://ru.wikipedia.org/wiki/%D0%AE%D0%BD%D0%B8%D0%BA%D0%BE%D0%B4>
4. Википедия locale.h - <https://ru.wikipedia.org/wiki/Locale.h>
5. YouTube: Гоша Дударь -
https://www.youtube.com/watch?v=hBJ4BWhP0OE&list=PL0lO_mIqDDFX2VcYQrDzrvYpzMVNexrp0&ab_channel=%D0%93%D0%BE%D1%88%D0%B0%D0%94%D1%83%D0%B4%D0%B0%D1%80%D1%8C
6. Metanit - <https://metanit.com/c/tutorial/5.1.php>
7. Code.c - <https://1w1s.github.io/Code.c/learning/DMA.html>
8. Материалы электронного курса на Moodle «Структурное программирование» и «Курсовой проект».

Листинг.

```
#include <stdio.h>
#include <wchar.h>
#include <locale.h>
#include <stdbool.h>
#include <wctype.h>
#include <stdlib.h>
#include <string.h>

#define INITIAL_SIZE 20

typedef struct {
    wchar_t *russian;
    wchar_t *english;
} WordPair;

typedef struct {
    WordPair *standardWords;
    int numStandardWords;
    int capacityStandardWords;
    WordPair *addedWords;
    int numAddedWords;
    int capacityAddedWords;
} Dictionary;

void initializeDictionary(Dictionary *dict) {
    dict->numStandardWords = 0;
    dict->capacityStandardWords = INITIAL_SIZE;
    dict->standardWords = malloc(dict->capacityStandardWords *
sizeof(WordPair));
    if (dict->standardWords == NULL) {
        wprintf(L"Ошибка выделения памяти.\n");
        exit(1);
    }

    dict->numAddedWords = 0;
    dict->capacityAddedWords = INITIAL_SIZE;
    dict->addedWords = malloc(dict->capacityAddedWords *
sizeof(WordPair));
    if (dict->addedWords == NULL) {
        wprintf(L"Ошибка выделения памяти.\n");
        exit(1);
    }
}
```



```

    }
}

// Функция для приведения первой буквы строки к верхнему регистру
void capitalize(wchar_t *str) {
    if (str && *str) {
        *str = towupper(*str);
    }
}

// Функция для приведения строки к нижнему регистру
void toLowerCase(wchar_t *str) {
    for (; *str; ++str) *str = towlower(*str);
}

// Функция для перевода слова
bool translateWord(wchar_t word[], Dictionary *dict, wchar_t *translation)
{
    toLowerCase(word); // Приводим входное слово к нижнему регистру

    for (int i = 0; i < dict->numStandardWords; i++) {
        wchar_t russianLower[50];
        wcscpy(russianLower, dict->standardWords[i].russian);
        toLowerCase(russianLower);

        if (wcscmp(russianLower, word) == 0) {
            wcscpy(translation, dict->standardWords[i].english);
            return true;
        }
    }

    for (int i = 0; i < dict->numAddedWords; i++) {
        wchar_t russianLower[50];
        wcscpy(russianLower, dict->addedWords[i].russian);
        toLowerCase(russianLower);

        if (wcscmp(russianLower, word) == 0) {
            wcscpy(translation, dict->addedWords[i].english);
            return true;
        }
    }
}

```

```

        return false;
    }

// Функция для перевода текста
void translateText(wchar_t text[], Dictionary *dict) {
    wchar_t *result = malloc(1024 * sizeof(wchar_t));
    result[0] = L'\0';

    wchar_t buffer[50];
    wchar_t *word = text;
    wchar_t *start = text;

    while (*word != L'\0') {
        if (iswspace(*word) || iswpunct(*word)) {
            wcsncpy(buffer, start, word - start);
            buffer[word - start] = L'\0';

            if (wcslen(buffer) > 0) {
                wchar_t translation[50];
                if (translateWord(buffer, dict, translation)) {
                    wcscat(result, translation);
                } else {
                    wcscat(result, buffer);
                }
            }

            wchar_t punct[2] = {*word, L'\0'};
            wcscat(result, punct);

            start = word + 1;
        }
        word++;
    }

    if (start < word) {
        wcsncpy(buffer, start, word - start);
        buffer[word - start] = L'\0';

        if (wcslen(buffer) > 0) {
            wchar_t translation[50];
            if (translateWord(buffer, dict, translation)) {
                wcscat(result, translation);
            }
        }
    }
}

```

```

        } else {
            wcscat(result, buffer);
        }
    }
}

wprintf(L"Переведенный текст: %ls\n", result);
free(result);
}

// Функция для добавления нового слова в словарь
void addWord(Dictionary *dict) {
    while (true) {
        if (dict->numAddedWords >= dict->capacityAddedWords) {
            dict->capacityAddedWords *= 2;
            dict->addedWords = realloc(dict->addedWords, dict->
capacityAddedWords * sizeof(WordPair));
            if (dict->addedWords == NULL) {
                wprintf(L"Ошибка выделения памяти.\n");
                exit(1);
            }
        }
    }

    wchar_t russian[50];
    wchar_t english[50];

    wprintf(L"Введите новое слово на русском (или 0 для выхода): ");
    fgetws(russian, 50, stdin);
    wchar_t *newline = wcschr(russian, L'\n');
    if (newline) *newline = L'\0';

    if (wcscmp(russian, L"0") == 0) break;

    toLowerCase(russian); // Приводим слово к нижнему регистру

    wprintf(L"Введите перевод на английском: ");
    fgetws(english, 50, stdin);
    newline = wcschr(english, L'\n');
    if (newline) *newline = L'\0';

    toLowerCase(english); // Приводим слово к нижнему регистру
    capitalize(russian); // Приводим первую букву к верхнему регистру

```

```

        capitalize(english); // Приводим первую букву к верхнему регистру

        dict->addedWords[dict->numAddedWords].russian =
malloc((wcslen(russian) + 1) * sizeof(wchar_t));
        dict->addedWords[dict->numAddedWords].english =
malloc((wcslen(english) + 1) * sizeof(wchar_t));

        wcscpy(dict->addedWords[dict->numAddedWords].russian, russian);
        wcscpy(dict->addedWords[dict->numAddedWords].english, english);

        dict->numAddedWords++;
        wprintf(L"Слово успешно добавлено в словарь.\n");
    }
}

// Функция для добавления нового текста в словарь
void addText(Dictionary *dict) {
    wchar_t text[500];
    wprintf(L"Введите текст на русском: ");
    fgetws(text, 500, stdin);
    wchar_t *newline = wcschr(text, L'\n');
    if (newline) *newline = L'\0';

    wchar_t buffer[50];
    wchar_t *word = text;
    wchar_t *start = text;

    while (*word != L'\0') {
        if (iswspace(*word) || iswpunct(*word)) {
            wcsncpy(buffer, start, word - start);
            buffer[word - start] = L'\0';

            if (wcslen(buffer) > 0) {
                wchar_t translation[50];
                if (!translateWord(buffer, dict, translation)) {
                    wprintf(L"Слово '%ls' не найдено в словаре.\n",
buffer);

                    wprintf(L"Введите перевод слова на английском: ");
                    wchar_t newEnglish[50];
                    fgetws(newEnglish, 50, stdin);
                    wchar_t *newline = wcschr(newEnglish, L'\n');
                    if (newline) *newline = L'\0';

```

```

        if (dict->numAddedWords >= dict->capacityAddedWords) {
            dict->capacityAddedWords *= 2;
            dict->addedWords = realloc(dict->addedWords, dict-
>capacityAddedWords * sizeof(WordPair));
            if (dict->addedWords == NULL) {
                wprintf(L"Ошибка выделения памяти.\n");
                exit(1);
            }
        }

        toLowerCase(buffer); // Приводим слово к нижнему
регистру перед добавлением
        toLowerCase(newEnglish); // Приводим перевод к нижнему
регистру перед добавлением
        capitalize(buffer); // Приводим первую букву к
верхнему регистру
        capitalize(newEnglish);

        dict->addedWords[dict->numAddedWords].russian =
malloc((wcslen(buffer) + 1) * sizeof(wchar_t));
        dict->addedWords[dict->numAddedWords].english =
malloc((wcslen(newEnglish) + 1) * sizeof(wchar_t));

        wcscpy(dict->addedWords[dict->numAddedWords].russian,
buffer);
        wcscpy(dict->addedWords[dict->numAddedWords].english,
newEnglish);

        dict->numAddedWords++;
        wprintf(L"Слово успешно добавлено в словарь.\n");
    }
}

    start = word + 1;
}
word++;
}

if (start < word) {
    wcsncpy(buffer, start, word - start);
    buffer[word - start] = L'\0';
}

```

```

        if (wcslen(buffer) > 0) {
            wchar_t translation[50];
            if (!translateWord(buffer, dict, translation)) {
                wprintf(L"Слово '%ls' не найдено в словаре.\n", buffer);
                wprintf(L"Введите перевод слова на английском: ");
                wchar_t newEnglish[50];
                fgetws(newEnglish, 50, stdin);
                wchar_t *newline = wcschr(newEnglish, L'\n');
                if (newline) *newline = L'\0';

                if (dict->numAddedWords >= dict->capacityAddedWords) {
                    dict->capacityAddedWords *= 2;
                    dict->addedWords = realloc(dict->addedWords, dict-
>capacityAddedWords * sizeof(WordPair));
                    if (dict->addedWords == NULL) {
                        wprintf(L"Ошибка выделения памяти.\n");
                        exit(1);
                    }
                }

                toLowerCase(buffer); // Приводим слово к нижнему регистру
перед добавлением
                toLowerCase(newEnglish); // Приводим перевод к нижнему
регистру перед добавлением
                capitalize(buffer); // Приводим первую букву к верхнему
регистру
                capitalize(newEnglish);

                dict->addedWords[dict->numAddedWords].russian =
malloc((wcslen(buffer) + 1) * sizeof(wchar_t));
                dict->addedWords[dict->numAddedWords].english =
malloc((wcslen(newEnglish) + 1) * sizeof(wchar_t));

                wcscpy(dict->addedWords[dict->numAddedWords].russian,
buffer);
                wcscpy(dict->addedWords[dict->numAddedWords].english,
newEnglish);

                dict->numAddedWords++;
                wprintf(L"Слово успешно добавлено в словарь.\n");
            }
        }
    }
}

```

```

}

// Функция для редактирования существующего слова
void editWord(Dictionary *dict) {
    wchar_t russian[50];
    wprintf(L"Введите слово на русском для редактирования: ");
    fgetws(russian, 50, stdin);
    wchar_t *newline = wcschr(russian, L'\n');
    if (newline) *newline = L'\0';

    toLowerCase(russian); // Приводим слово к нижнему регистру

    for (int i = 0; i < dict->numStandardWords; i++) {
        wchar_t russianLower[50];
        wcscpy(russianLower, dict->standardWords[i].russian);
        toLowerCase(russianLower);

        if (wcscmp(russianLower, russian) == 0) {
            wchar_t english[50];
            wprintf(L"Введите новый перевод на английском: ");
            fgetws(english, 50, stdin);
            newline = wcschr(english, L'\n');
            if (newline) *newline = L'\0';

            toLowerCase(english); // Приводим слово к нижнему регистру
            capitalize(english); // Приводим первую букву к верхнему
регистру

            free(dict->standardWords[i].english);
            dict->standardWords[i].english = malloc((wcslen(english) + 1)
* sizeof(wchar_t));
            wcscpy(dict->standardWords[i].english, english);
            wprintf(L"Слово успешно отредактировано.\n");
            return;
        }
    }

    for (int i = 0; i < dict->numAddedWords; i++) {
        wchar_t russianLower[50];
        wcscpy(russianLower, dict->addedWords[i].russian);
        toLowerCase(russianLower);
    }
}

```

```

        if (wcscmp(russianLower, russian) == 0) {
            wchar_t english[50];
            wprintf(L"Введите новый перевод на английском: ");
            fgetws(english, 50, stdin);
            newline = wcschr(english, L'\n');
            if (newline) *newline = L'\0';

            toLowerCase(english); // Приводим слово к нижнему регистру
            capitalize(english); // Приводим первую букву к верхнему
регистру

            free(dict->addedWords[i].english);
            dict->addedWords[i].english = malloc((wcslen(english) + 1) *
sizeof(wchar_t));
            wcscpy(dict->addedWords[i].english, english);
            wprintf(L"Слово успешно отредактировано.\n");
            return;
        }
    }

    wprintf(L"Слово не найдено в словаре.\n");
}

// Функция для удаления слова из словаря
void deleteWord(Dictionary *dict) {
    wchar_t russian[50];
    wprintf(L"Введите слово на русском для удаления: ");
    fgetws(russian, 50, stdin);
    wchar_t *newline = wcschr(russian, L'\n');
    if (newline) *newline = L'\0';

    toLowerCase(russian); // Приводим слово к нижнему регистру

    for (int i = 0; i < dict->numStandardWords; i++) {
        wchar_t russianLower[50];
        wcscpy(russianLower, dict->standardWords[i].russian);
        toLowerCase(russianLower);

        if (wcscmp(russianLower, russian) == 0) {
            free(dict->standardWords[i].russian);
            free(dict->standardWords[i].english);
            for (int j = i; j < dict->numStandardWords - 1; j++) {

```



```

        dict->standardWords[j] = dict->standardWords[j + 1];
    }
    dict->numStandardWords--;
    wprintf(L"Слово успешно удалено из словаря.\n");
    return;
}
}

for (int i = 0; i < dict->numAddedWords; i++) {
    wchar_t russianLower[50];
    wcsncpy(russianLower, dict->addedWords[i].russian);
    toLowerCase(russianLower);

    if (wcscmp(russianLower, russian) == 0) {
        free(dict->addedWords[i].russian);
        free(dict->addedWords[i].english);
        for (int j = i; j < dict->numAddedWords - 1; j++) {
            dict->addedWords[j] = dict->addedWords[j + 1];
        }
        dict->numAddedWords--;
        wprintf(L"Слово успешно удалено из словаря.\n");
        return;
    }
}

wprintf(L"Слово не найдено в словаре.\n");
}

// Функция для отображения всего словаря
void displayDictionary(Dictionary *dict) {
    for (int i = 0; i < dict->numStandardWords; i++) {
        wprintf(L"%ls: %ls\n", dict->standardWords[i].russian, dict->standardWords[i].english);
    }

    for (int i = 0; i < dict->numAddedWords; i++) {
        wprintf(L"%ls: %ls\n", dict->addedWords[i].russian, dict->addedWords[i].english);
    }
}

// Функция для сохранения словаря в файл

```

```

void saveDictionary(Dictionary *dict) {
    wchar_t filename[100];
    wprintf(L"Введите имя файла для сохранения: ");
    fgetws(filename, 100, stdin);
    wchar_t *newline = wcschr(filename, L'\n');
    if (newline) *newline = L'\0';

    FILE *file = _wopen(filename, L"w");
    if (file == NULL) {
        wprintf(L"Ошибка открытия файла.\n");
        return;
    }

    fwprintf(file, L"STANDARD_WORDS\n");
    for (int i = 0; i < dict->numStandardWords; i++) {
        fwprintf(file, L"%ls %ls\n", dict->standardWords[i].russian, dict->standardWords[i].english);
    }

    fwprintf(file, L"ADDED_WORDS\n");
    for (int i = 0; i < dict->numAddedWords; i++) {
        fwprintf(file, L"%ls %ls\n", dict->addedWords[i].russian, dict->addedWords[i].english);
    }

    fclose(file);
    wprintf(L"Словарь сохранен в файл %ls.\n", filename);
}

// Функция для загрузки словаря из файла
void loadDictionary(Dictionary *dict) {
    wchar_t filename[100];
    wprintf(L"Введите имя файла для загрузки: ");
    fgetws(filename, 100, stdin);
    wchar_t *newline = wcschr(filename, L'\n');
    if (newline) *newline = L'\0';

    FILE *file = _wopen(filename, L"r");
    if (file == NULL) {
        wprintf(L"Ошибка открытия файла.\n");
        return;
    }
}

```

```

// Освобождаем текущий словарь
for (int i = 0; i < dict->numStandardWords; i++) {
    free(dict->standardWords[i].russian);
    free(dict->standardWords[i].english);
}
free(dict->standardWords);

for (int i = 0; i < dict->numAddedWords; i++) {
    free(dict->addedWords[i].russian);
    free(dict->addedWords[i].english);
}
free(dict->addedWords);

// Инициализируем новый словарь
initializeDictionary(dict);

wchar_t russian[50];
wchar_t english[50];
wchar_t buffer[100];
bool readingStandardWords = true;

while (fgetws(buffer, 100, file)) {
    if (wcscmp(buffer, L"STANDARD_WORDS\n") == 0) {
        readingStandardWords = true;
        continue;
    }
    if (wcscmp(buffer, L"ADDED_WORDS\n") == 0) {
        readingStandardWords = false;
        continue;
    }

    if (swscanf(buffer, L"%49ls %49ls", russian, english) == 2) {
        toLowerCase(russian); // Приводим слово к нижнему регистру
        перед добавлением
        toLowerCase(english); // Приводим перевод к нижнему регистру
        перед добавлением
        capitalize(russian); // Приводим первую букву к верхнему
        регистру
        capitalize(english); // Приводим первую букву к верхнему
        регистру
    }
}

```

```

        if (readingStandardWords) {
            if (dict->numStandardWords >= dict->capacityStandardWords)
            {
                dict->capacityStandardWords *= 2;
                dict->standardWords = realloc(dict->standardWords,
dict->capacityStandardWords * sizeof(WordPair));
                if (dict->standardWords == NULL) {
                    wprintf(L"Ошибка выделения памяти.\n");
                    exit(1);
                }
            }
            dict->standardWords[dict->numStandardWords].russian =
malloc((wcslen(russian) + 1) * sizeof(wchar_t));
            dict->standardWords[dict->numStandardWords].english =
malloc((wcslen(english) + 1) * sizeof(wchar_t));

            wcscpy(dict->standardWords[dict-
>numStandardWords].russian, russian);
            wcscpy(dict->standardWords[dict-
>numStandardWords].english, english);
            dict->numStandardWords++;
        } else {
            if (dict->numAddedWords >= dict->capacityAddedWords) {
                dict->capacityAddedWords *= 2;
                dict->addedWords = realloc(dict->addedWords, dict-
>capacityAddedWords * sizeof(WordPair));
                if (dict->addedWords == NULL) {
                    wprintf(L"Ошибка выделения памяти.\n");
                    exit(1);
                }
            }
            dict->addedWords[dict->numAddedWords].russian =
malloc((wcslen(russian) + 1) * sizeof(wchar_t));
            dict->addedWords[dict->numAddedWords].english =
malloc((wcslen(english) + 1) * sizeof(wchar_t));

            wcscpy(dict->addedWords[dict->numAddedWords].russian,
russian);
            wcscpy(dict->addedWords[dict->numAddedWords].english,
english);
            dict->numAddedWords++;
        }
    }
}

```

```

    }
}

fclose(file);
wprintf(L"Словарь загружен из файла %ls.\n", filename);
}

// Освобождение памяти
void freeDictionary(Dictionary *dict) {
    for (int i = 0; i < dict->numStandardWords; i++) {
        free(dict->standardWords[i].russian);
        free(dict->standardWords[i].english);
    }
    free(dict->standardWords);

    for (int i = 0; i < dict->numAddedWords; i++) {
        free(dict->addedWords[i].russian);
        free(dict->addedWords[i].english);
    }
    free(dict->addedWords);
}

int main() {
    setlocale(LC_ALL, ".866");
    setlocale(LC_CTYPE, "ru_RU.UTF-8");

    Dictionary dict;
    initializeDictionary(&dict);

    // Инициализация словаря
    wchar_t *initialWords[10][2] = {
        {L"Привет", L"Hello"},
        {L"Пока", L"Goodbye"},
        {L"Спасибо", L"Thank you"},
        {L"Пожалуйста", L"Please"},
        {L"Да", L"Yes"},
        {L"Нет", L"No"},
        {L"Извините", L"Sorry"},
        {L"Утро", L"Morning"},
        {L"Вечер", L"Evening"},
        {L"Ночь", L"Night"}
    };
};

```

```

        for (int i = 0; i < 10; i++) {
            dict.standardWords[dict.numStandardWords].russian =
                malloc((wcslen(initialWords[i][0]) + 1) * sizeof(wchar_t));
            dict.standardWords[dict.numStandardWords].english =
                malloc((wcslen(initialWords[i][1]) + 1) * sizeof(wchar_t));
            wcscpy(dict.standardWords[dict.numStandardWords].russian,
                initialWords[i][0]);
            wcscpy(dict.standardWords[dict.numStandardWords].english,
                initialWords[i][1]);
            dict.numStandardWords++;
        }

    int choice;
    while (true) {
        wprintf(L"Меню:\n");
        wprintf(L"1. Перевести слово\n");
        wprintf(L"2. Перевести текст\n");
        wprintf(L"3. Добавить новое слово\n");
        wprintf(L"4. Добавить новый текст для перевода\n");
        wprintf(L"5. Редактировать существующее слово\n");
        wprintf(L"6. Удалить слово\n");
        wprintf(L"7. Отобразить словарь\n");
        wprintf(L"8. Сохранить словарь в файл\n");
        wprintf(L"9. Загрузить словарь из файла\n");
        wprintf(L"0. Выход\n");
        wprintf(L"Выберите опцию: ");
        wscanf(L"%d", &choice);
        getchar(); // Удаление символа новой строки из входного буфера

        switch (choice) {
            case 1: {
                wchar_t input[50];
                while (true) {
                    wprintf(L"Введите слово на русском (для выхода введите
'0'): ");

                    fgetws(input, 50, stdin);
                    wchar_t *newline = wcschr(input, L'\n');
                    if (newline) *newline = L'\0';

                    if (wcscmp(input, L"0") == 0) break;
                }
            }
        }
    }

```

```

        wchar_t translation[50];
        if (translateWord(input, &dict, translation)) {
            wprintf(L"Перевод: %ls\n", translation);
        } else {
            wprintf(L"Слово не найдено в словаре.\n");
        }
    }
    break;
}
case 2: {
    wchar_t text[500];
    wprintf(L"Введите текст на русском: ");
    fgetws(text, 500, stdin);
    wchar_t *newline = wcschr(text, L'\n');
    if (newline) *newline = L'\0';

    translateText(text, &dict);
    break;
}
case 3:
    addWord(&dict);
    break;
case 4:
    addText(&dict);
    break;
case 5:
    editWord(&dict);
    break;
case 6:
    deleteWord(&dict);
    break;
case 7:
    displayDictionary(&dict);
    break;
case 8:
    saveDictionary(&dict);
    break;
case 9:
    loadDictionary(&dict);
    break;
case 0:
    freeDictionary(&dict);

```

```
        return 0;
    default:
        wprintf(L"Некорректный выбор. Пожалуйста, попробуйте
снова.\n");
    }
}
}
```