УДК 004

Создание преобразователя устной речи в текст на языке программирования C#

Ульянов Егор Андреевич
Приамурский государственный университет имени Шолом-Алейхема
Студент

Аннотация
В данной статье представлен обзор процесса создания преобразователя устной речи в текст на языке программирования C#, что позволяет значительно упростить взаимодействие пользователя с компьютерными системами. Результаты работы могут быть применены в различных областях, включая автоматизированный ввод данных, помощь людям с ограниченными возможностями и улучшение пользовательского опыта в целом.
Ключевые слова: преобразователь, C#, Visual Studio

Creating an oral speech to text converter in the C programming language#

Ulianov Egor Andreevich
Sholom-Aleichem Priamursky State University
Student

Abstract
This article provides an overview of the process of creating an oral speech to text converter in the C# programming language, which greatly simplifies user interaction with computer systems. The results of the work can be applied in various fields, including automated data entry, helping people with disabilities and improving the user experience in general.
Keywords: converter, C#, Visual Studio

В современном мире технологии распознавания речи играют ключевую роль в улучшении доступности и интерактивности цифровых устройств и сервисов. Программы преобразующие устную речь в текст, как никогда актуальны, поскольку представляют собой значительный вклад в развитие инклюзивных технологий, позволяющих людям с нарушениями слуха или речи взаимодействовать с компьютерными системами более эффективно. Кроме того, такие системы могут быть использованы для автоматизации и оптимизации рабочих процессов, например, в области документирования или обслуживания клиентов. Разработка на языке C# делает преобразователь легко интегрируемым в широкий спектр приложений.

Целью данной статьи является создание преобразователя устной речи в текст в среде разработки «Visual Studio» на языке программирования C#.

В своей работе Н. Н. Додобоев, О. И. Кукарцева, Я. А. Тынченко рассмотрели вопросы появления различных языков программирования (в частности C#), определения особенностей этих языков, а также составления основных видов и классификаций языков программирования [1]. З. С. Магомадова рассмотрела языки программирования высокого уровня, особенности, недостатки и сложности в изучении, а также описала несколько легких алгоритмов [2]. В статье Е. О. Шмигириловой освещается прогресс в области распознавания речи, ключевой технологии, которая становится всё более важной в эпоху цифровизации. Автор фокусируются на разработке алгоритмов, способных точно и быстро преобразовывать устную речь в текстовый формат [3]. В работе Э. Г. Амирасланова, С. Э. Сараджишвили, Т. В. Леонтьевой рассматривается вопрос создания системы голосового управления, сравнение современных методов распознавания, и выявление наиболее эффективных и менее ресурсоемких подходов [4].

Создаем проект «Windows Forms App» и называем необходимым именем см. рисунок 1.

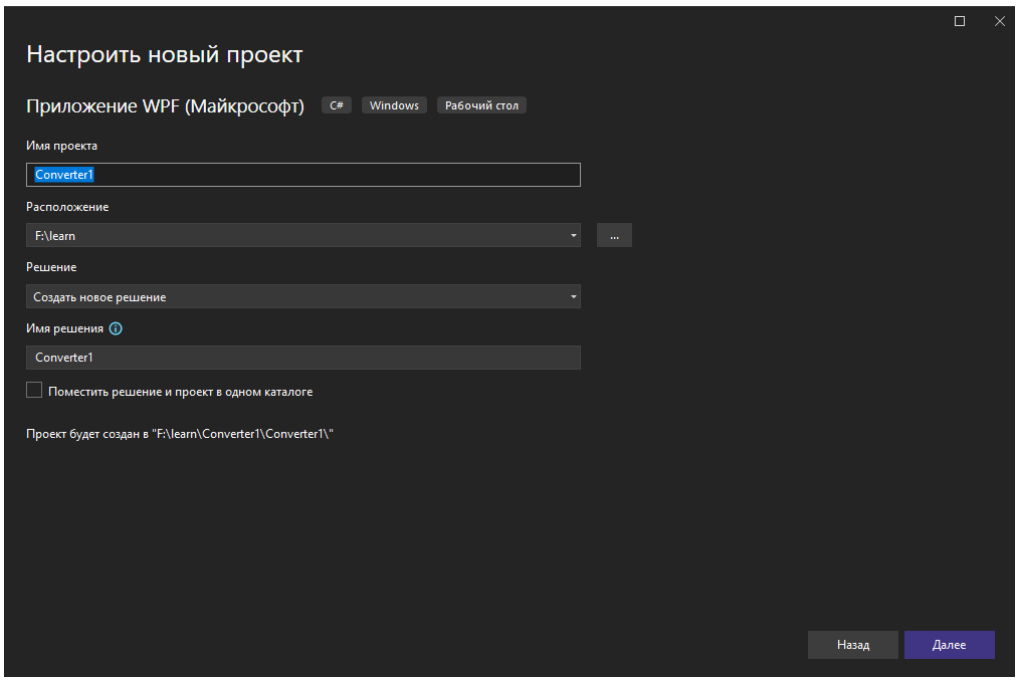


Рис. 1. Создание проекта

Для начала необходимо добавить библиотеку System.Speech в сборку в GAC. (см. рисунке 2).

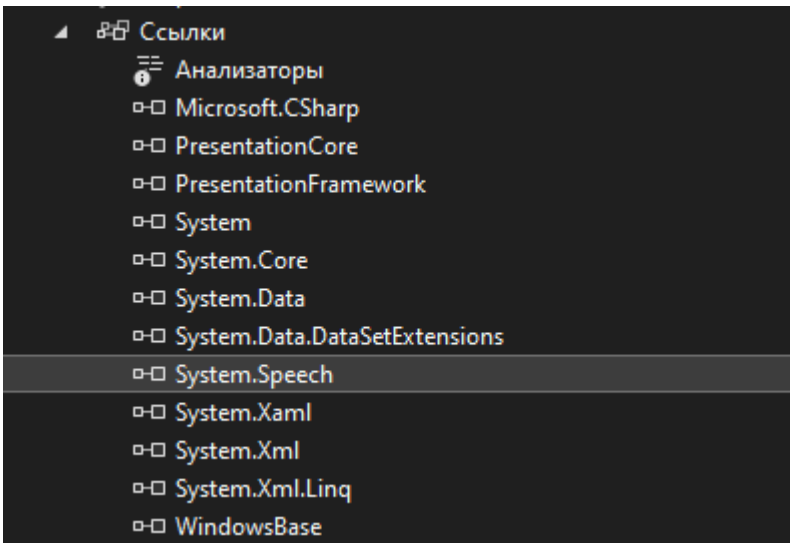


Рис. 2. Добавление библиотеки System.Speech

Расставляем компоненты в необходимом порядке (см. рисунок 3).

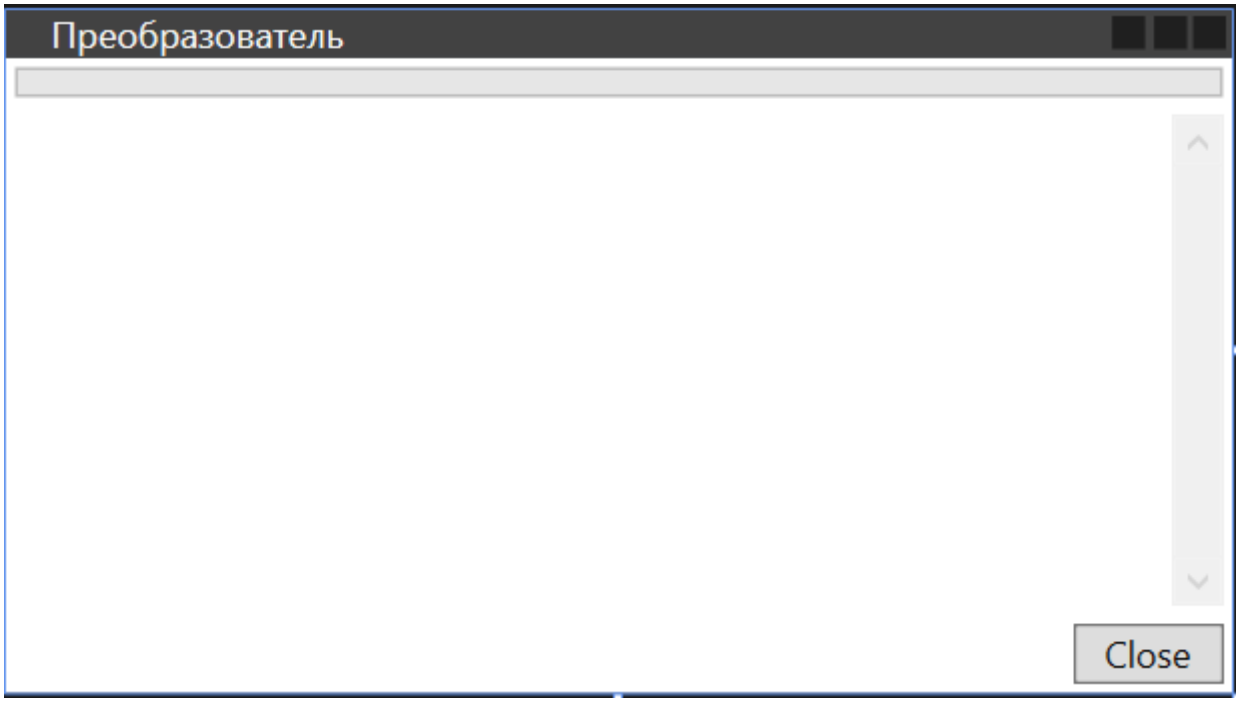


Рис. 3. Расположение компонентов

Переходим к написанию кода. Пространство имен SpeechToText - определяет область, в которой находится класс MainWindow, отвечающий за основное окно приложения. Класс MainWindow - является частичным классом и наследуется от Window, что указывает на то, что он представляет собой окно в графическом интерфейсе пользователя. speechRecognitionEngine будет использоваться для распознавания речи. Words - список объектов Word, используется для хранения распознанных слов. Конструктор MainWindow- инициализирует компоненты интерфейса с помощью метода InitializeComponent(). Создает экземпляр SpeechRecognitionEngine для английского языка ("en-US"), подписывается на события AudioLevelUpdated

и SpeechRecognized, которые будут вызываться при изменении уровня звука и успешном распознавании речи соответственно, вызывает метод loadGrammarAndCommands(), который, вероятно, загружает грамматику и команды для распознавания. А также устанавливает аудиоустройство по умолчанию для speechRecognitionEngine и начинает асинхронное распознавание речи в режиме RecognizeMode.Multiple, что позволяет распознавать несколько команд в течение одной сессии (см. рисунок 4).

```
namespace SpeechToText
{

    Ссылок: 3
    public partial class MainWindow : Window
    {
        #region locals


        SpeechRecognitionEngine speechRecognitionEngine = null;


        List<Word> words = new List<Word>();

        #endregion

        #region ctor

        Ссылок: 0
        public MainWindow()
        {
            InitializeComponent();

            try
            {

                speechRecognitionEngine = createSpeechEngine("de-DE");


                speechRecognitionEngine.AudioLevelUpdated += new EventHandler<AudioLevelUpdatedEventArgs>(engine_AudioLevelUpdated);
                speechRecognitionEngine.SpeechRecognized += new EventHandler<SpeechRecognizedEventArgs>(engine_SpeechRecognized);

                loadGrammarAndCommands();


                speechRecognitionEngine.SetInputToDefaultAudioDevice();

                speechRecognitionEngine.RecognizeAsync(RecognizeMode.Multiple);
            }
            catch (Exception ex)
            {
                MessageBox.Show(ex.Message, "Voice recognition failed");
            }
        }
```

Рис. 4. Класс MainWindow

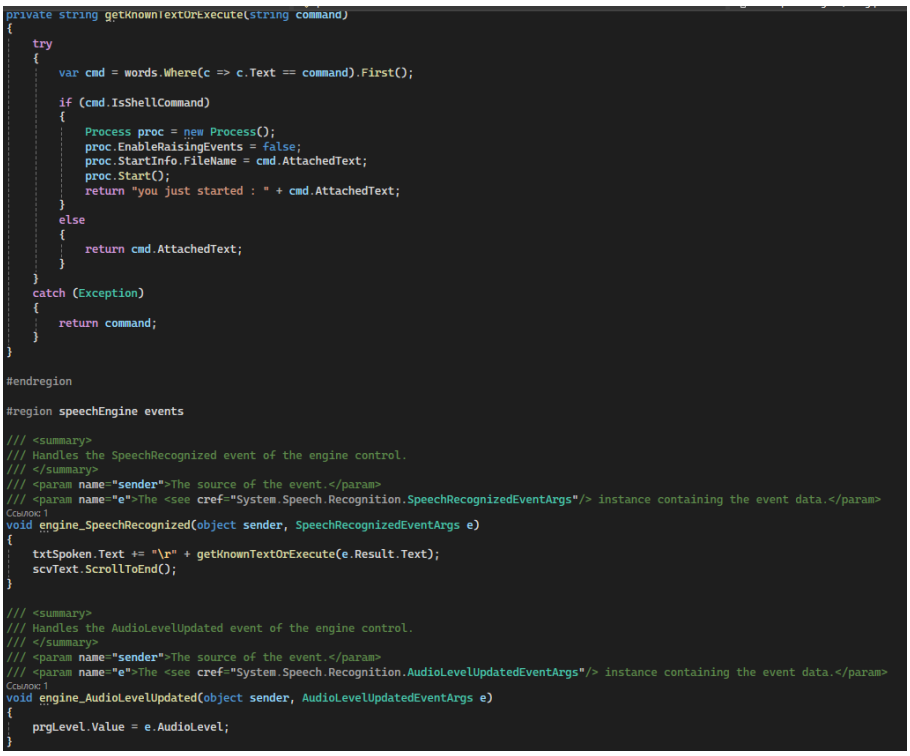
Создаем метод createSpeechEngine, принимающий строку preferredCulture, которая представляет предпочтительный культурный контекст (например, язык и регион). Итерирует через установленные распознаватели речи на машине, ищет совпадение с preferredCulture. Если находит соответствующий RecognizerInfo, создает новый SpeechRecognitionEngine с этой конфигурацией. Если соответствующая культура не найдена, выводит сообщение об ошибке и создает SpeechRecognitionEngine с первой доступной культурой. Возвращает экземпляр SpeechRecognitionEngine. Также создаем метод loadGrammarAndCommands с такими функциями как: создание объекта Choices, который будет хранить возможные варианты распознаваемой речи; чтение строки из файла example.txt, который находится в текущем рабочем каталоге; игнорирование строки, начинающиеся с -- или пустые строки; разделения каждой строки на части по символу |; добавление каждого

распознанного слова в список words и в Choices; создание грамматики wordsList из texts и загрузка в speechRecognitionEngine см. рисунок 5.

```
private SpeechRecognitionEngine createSpeechEngine(string preferredCulture)
{
    foreach (RecognizerInfo config in SpeechRecognitionEngine.InstalledRecognizers())
    {
        if (config.Culture.ToString() == preferredCulture)
        {
            speechRecognitionEngine = new SpeechRecognitionEngine(config);
            break;
        }
    }

    if (speechRecognitionEngine == null)
    {
        MessageBox.Show("The desired culture is not installed on this machine, the speech-engine will continue using "
            + SpeechRecognitionEngine.InstalledRecognizers()[0].Culture.ToString() + " as the default culture.",
            "Culture " + preferredCulture + " not found!");
        speechRecognitionEngine = new SpeechRecognitionEngine(SpeechRecognitionEngine.InstalledRecognizers()[0]);
    }

    return speechRecognitionEngine;
}

Ссылок: 1
private void loadGrammarAndCommands()
{
    try
    {
        Choices texts = new Choices();
        string[] lines = File.ReadAllLines(Environment.CurrentDirectory + "\\example.txt");
        foreach (string line in lines)
        {

            if (line.StartsWith("--") || line == String.Empty) continue;

            var parts = line.Split(new char[] { '|' });

            words.Add(new Word() { Text = parts[0], AttachedText = parts[1], IsShellCommand = (parts[2] == "true") });

            // add the text to the known choices of speechengine
            texts.Add(parts[0]);
        }
        Grammar wordsList = new Grammar(new GrammarBuilder(texts));
        speechRecognitionEngine.LoadGrammar(wordsList);
    }
    catch (Exception ex)
    {
        throw ex;
    }
}
```

Рис. 5. Методы createSpeechEngine и loadGrammarAndCommands

Теперь создаем метод getKnownTextOrExecute и обработчик событий для распознавания речи см. рисунок 6.



```
private string getKnownTextOrExecute(string command)
{
    try
    {
        var cmd = words.Where(c => c.Text == command).First();

        if (cmd.IsShellCommand)
        {
            Process proc = new Process();
            proc.EnableRaisingEvents = false;
            proc.StartInfo.FileName = cmd.AttachedText;
            proc.Start();
            return "you just started : " + cmd.AttachedText;
        }
        else
        {
            return cmd.AttachedText;
        }
    }
    catch (Exception)
    {
        return command;
    }
}

#endregion

#region speechEngine events

/// <summary>
/// Handles the SpeechRecognized event of the engine control.
/// </summary>
/// <param name="sender">The source of the event.</param>
/// <param name="e">The <see cref="System.Speech.Recognition.SpeechRecognizedEventArgs"/> instance containing the event data.</param>
Ссылок: 1
void engine_SpeechRecognized(object sender, SpeechRecognizedEventArgs e)
{
    txtSpoken.Text += "\r" + getKnownTextOrExecute(e.Result.Text);
    scvText.ScrollToEnd();
}

/// <summary>
/// Handles the AudioLevelUpdated event of the engine control.
/// </summary>
/// <param name="sender">The source of the event.</param>
/// <param name="e">The <see cref="System.Speech.Recognition.AudioLevelUpdatedEventArgs"/> instance containing the event data.</param>
Ссылок: 1
void engine_AudioLevelUpdated(object sender, AudioLevelUpdatedEventArgs e)
{
    prgLevel.Value = e.AudioLevel;
}
```

Рис. 6. Метод getKnownTextOrExecute

Далее необходимо доработать графический пользовательский интерфейс (GUI), для этого понадобится метод Window_Closing. Этот метод является обработчиком события закрытия окна. Внутри метода вызывается RecognizeAsyncStop для остановки асинхронного распознавания речи. Затем вызывается Dispose для освобождения ресурсов, связанных с speechRecognitionEngine. Этот метод гарантирует, что при закрытии окна приложения ресурсы движка распознавания речи будут корректно освобождены. Метод Button_Click обрабатывает событие нажатия на кнопку. Когда кнопка нажата, вызывается метод Close, который закрывает текущее окно. Это стандартный способ добавления функциональности кнопке в приложениях WPF для закрытия окна см. рисунок 7.

```
private void Window_Closing(object sender, System.ComponentModel.CancelEventArgs e)
{

    speechRecognitionEngine.RecognizeAsyncStop();

    speechRecognitionEngine.Dispose();
}

#endregion

#region GUI events


private void Button_Click(object sender, RoutedEventArgs e)
{
    this.Close();
}
```

Рис. 7. Методы Window_Closing и Button_Click

Проверяем работу программу см. рисунок 8.

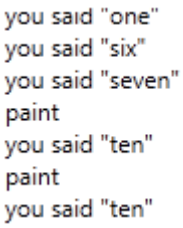


Рис. 8. Распознавание речи

В заключении, разработка преобразователя устной речи в текст на языке C# открывает новые горизонты для интерактивного программного обеспечения. Это не только демонстрирует возможности современных технологий распознавания речи, но и подчеркивает важность создания доступных и удобных инструментов для широкого круга пользователей. Применение кода на C# позволяют достичь высокой точности и скорости обработки данных, что делает преобразователь незаменимым помощником во многих сферах, включая образование, бизнес и развлечения.

Библиографический список

1. Додобоев Н. Н., Кукарцева О. И., Тынченко Я. А. Современные языки программирования // Современные технологии: актуальные вопросы, достижения и инновации. 2014. №5. С. 81-85.
2. Магомадова З. С. Языки программирования высокого уровня // Разработка и применение наукоёмких технологий в эпоху глобальных трансформаций. 2020. №8. С. 94-96.
3. Шмигирилова, Е. О. Распознование речи // Проблемы энергетики, природопользования, безопасности жизнедеятельности и экологии: Сборник материалов студенческой научно-практической конференции, Брянск, 12 апреля 2022 года. Брянск: Брянский государственный аграрный университет, 2022. С. 581-586.
4. Амирасланов, Э. Г. Улучшенный метод распознования речи для разработки системы голосового управления // Современная наука: актуальные проблемы теории и практики. Серия: Естественные и технические науки. 2023. № 4-2. С. 42-45.