Министерство образования Республики Беларусь

Учреждение образования

«Брестский государственный технический университет»

Кафедра ИИТ

Лабораторная работа №7-8

По дисциплине: «Естественно-языковой интерфейс ИС»

# Тема: «Разработка систем анализа речи», «Разработка систем синтеза речи»

Выполнил:

Студент 4 курса

Группы ИИ-21

Ливтинюк Т. В.

Проверила:

Якимук А. В.

Брест 2024

**Цель:** освоить на практике основные принципы создания систем анализа и синтеза речи. 

**Ход работы:**

 Ввод текста, или копирование через буфер, или поддержка указателя мыши в

любом другом приложении (например, на html-страничке).

 Воспроизведение сгенерированного речевого сигнала для введенного текста.

 Настройки (выбор) голоса диктора, темпа, громкости и др. параметров для чтения

 описание структуры разработанной системы, технологий и методик;

 основные алгоритмы и принципы реализации компонент системы (блок-схемы

алгоритмов);

 результаты тестирования;

 результаты анализа полученных данных, и предложения по улучшению работы

системы.

import tkinter as tk

from tkinter import ttk

import speech\_recognition as sr

import threading

class SpeechApp:

def \_\_init\_\_(self, root):

self.root = root

self.root.title("Приложение распознавания речи")

self.label = tk.Label(root, text="Нажмите кнопку и говорите:", font=("Arial", 16))

self.label.pack(pady=20)

self.result\_text = tk.Text(root, height=10, width=50)

self.result\_text.pack(pady=20)

# Добавляем выпадающий список для выбора языка

self.language\_var = tk.StringVar(value='ru-RU') # По умолчанию русский

self.language\_label = tk.Label(root, text="Выберите язык:", font=("Arial", 12))

self.language\_label.pack(pady=5)

self.language\_combobox = ttk.Combobox(root, textvariable=self.language\_var, values=[

'ru-RU', 'en-US', 'de-DE', 'fr-FR'])

self.language\_combobox.pack(pady=5)

self.recognize\_button = tk.Button(root, text="Распознать речь", command=self.start\_recognition\_thread, font=("Arial", 12))

self.recognize\_button.pack(pady=20)

def start\_recognition\_thread(self):

# Запуск функции распознавания речи в отдельном потоке

threading.Thread(target=self.recognize\_speech).start()

def recognize\_speech(self):

recognizer = sr.Recognizer()

recognizer.pause\_threshold = 1

with sr.Microphone() as source:

self.label.config(text="Слушаю...")

# self.root.update() # Обновляем интерфейс, чтобы метка изменилась

#recognizer.adjust\_for\_ambient\_noise(source, duration=1)

try:

audio = recognizer.listen(source, timeout=3, phrase\_time\_limit=12)

except sr.WaitTimeoutError as e:

self.label.config(text=f"Не удалось получить результаты; {e}.")

try:

language = self.language\_var.get() # Получаем выбранный язык

text = recognizer.recognize\_google(audio, language=language)

print(text)

self.result\_text.delete(1.0, tk.END) # Очистить предыдущий текст

self.result\_text.insert(tk.END, text) # Вставить распознанный текст

self.label.config(text="Нажмите кнопку и говорите:")

# Автоматическая реакция на определенные фразы

self.respond\_to\_command(text)

except sr.UnknownValueError:

self.label.config(text="Не удалось распознать аудио.")

except sr.RequestError as e:

self.label.config(text=f"Не удалось получить результаты; {e}.")

except sr.WaitTimeoutError as e:

self.label.config(text=f"Не удалось получить результаты; {e}.")

def respond\_to\_command(self, text):

# Реакция на команды

if "привет" in text.lower():

self.label.config(text="Привет! Как я могу помочь?")

elif "стоп" in text.lower():

self.label.config(text="А я что машина?")

elif "как дела" in text.lower():

self.label.config(text="У меня все хорошо, спасибо!")

if \_\_name\_\_ == '\_\_main\_\_':

root = tk.Tk()

app = SpeechApp(root)

root.mainloop()  
import tkinter as tk

from tkinter import ttk

import pyttsx3

class TTSApp:

def \_\_init\_\_(self, root):

self.root = root

self.root.title("Система синтеза речи")

self.engine = pyttsx3.init()

voices = self.engine.getProperty('voices')

for index, voice in enumerate(voices):

print(f"Voice {index}: {voice.name}")

self.label = tk.Label(root, text="Введите текст:", font=("Arial", 16))

self.label.pack(pady=10)

self.text\_entry = tk.Text(root, height=10, width=50)

self.text\_entry.pack(pady=10)

self.speak\_button = tk.Button(root, text="ПРОИЗНЕСИТЬ", command=self.speak\_text, font=("Arial", 12))

self.speak\_button.pack(pady=10)

self.voice\_label = tk.Label(root, text="Выберите голос:", font=("Arial", 12))

self.voice\_label.pack(pady=5)

self.voices = self.engine.getProperty('voices')

self.voice\_var = tk.StringVar(value=self.voices[0].id) # Установка первого голоса по умолчанию

self.voice\_combobox = ttk.Combobox(root, textvariable=self.voice\_var, values=[voice.name for voice in self.voices])

self.voice\_combobox.pack(pady=5)

self.speed\_label = tk.Label(root, text="Установите скорость:", font=("Arial", 12))

self.speed\_label.pack(pady=5)

self.speed\_scale = tk.Scale(root, from\_=50, to=300, orient=tk.HORIZONTAL, label="Скорость (слов в минуту)", length=400)

self.speed\_scale.set(150) # Установка значения по умолчанию

self.speed\_scale.pack(pady=10)

self.volume\_label = tk.Label(root, text="Установите громкость:", font=("Arial", 12))

self.volume\_label.pack(pady=5)

self.volume\_scale = tk.Scale(root, from\_=0.0, to=1.0, resolution=0.1, orient=tk.HORIZONTAL, label="Громкость", length=400)

self.volume\_scale.set(1.0) # Установка значения по умолчанию

self.volume\_scale.pack(pady=10)

def speak\_text(self):

text = self.text\_entry.get("1.0", tk.END).strip() # Получаем текст

self.engine.setProperty('voice', self.get\_selected\_voice\_id()) # Устанавливаем выбранный голос

self.engine.setProperty('rate', self.speed\_scale.get()) # Устанавливаем скорость

self.engine.setProperty('volume', self.volume\_scale.get()) # Устанавливаем громкость

self.engine.say(text) # Произносим текст

self.engine.runAndWait() # Ждем завершения

def get\_selected\_voice\_id(self):

selected\_voice\_name = self.voice\_var.get() # Получаем выбранное имя голоса

for voice in self.voices:

if voice.name == selected\_voice\_name:

return voice.id # Возвращаем ID выбранного голоса

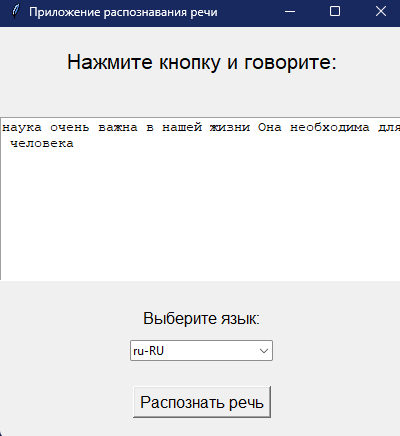
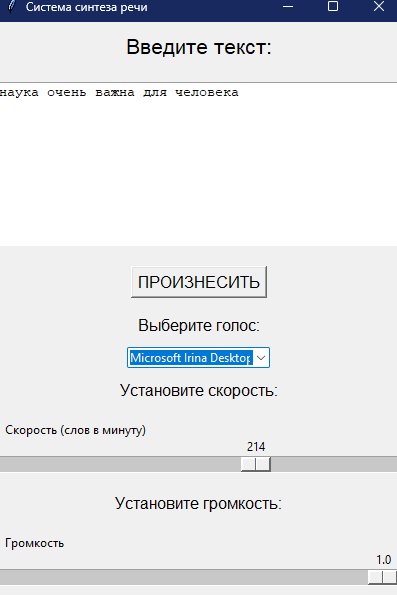
return self.voices[0].id # Если ничего не найдено, возвращаем первый голос

if \_\_name\_\_ == '\_\_main\_\_':

root = tk.Tk()

app = TTSApp(root)

root.mainloop()

****

**Вывод:** в ходе выполнения лабораторной работы освоил на практике основные принципы создания систем анализа и синтеза речи.