main.py

Необхідні бібліотеки

```
import numpy as np
import tkinter as tk
from tkinter import messagebox
```

`numpy` -- для числових обчислень

`tkinter` -- для створення графічного інтерфейсу

Визначення трапецієподібної функції належності

```
# Трапецієвидна функцій належності
def trapmf(x, a, b, c, d):
    if a < x < b:
        return (x - a) / (b - a)
    elif b <= x <= c:
        return 1
    elif c < x < d:
        return (x - d) / (c - d)
    else:
        return 0
```

Ця функція визначає трапецієподібну функцію належності, яка використовується для фазифікації вхідних значень. Вона повертає ступінь належності вхідного значення `x` до нечіткої множини, визначеної параметрами `a`, `b`, `c` і `d`.

Визначення функцій належності для різних показників

```
# Визначення функцій належності для систолічного тиску
def systolic_func(value):
    data = {'low': trapmf(value, a=80, b=80, c=90, d=110),
            'normal': trapmf(value, a=90, b=120, c=120, d=140),
            'high': trapmf(value, a=130, b=160, c=180, d=180)}
    return data
```

```
# Визначення функцій належності для діастолічного тиску
def diastolic_func(value):
    data = {'low': trapmf(value, a=50, b=50, c=60, d=80),
            'normal': trapmf(value, a=60, b=80, c=80, d=100),
            'high': trapmf(value, a=80, b=100, c=120, d=120)}
    return data

# Визначення функцій належності для пульсу
def pulse_func(value):
    data = {'low': trapmf(value, a=40, b=40, c=50, d=70),
            'normal': trapmf(value, a=50, b=70, c=70, d=90),
            'high': trapmf(value, a=80, b=120, c=180, d=180)}
    return data

# Визначення функцій належності для оцінки артеріального тиску
def blood_pressure_func(value):
    data = {'low': trapmf(value, a=0, b=0, c=20, d=50),
            'normal': trapmf(value, a=20, b=50, c=50, d=80),
            'high': trapmf(value, a=50, b=80, c=100, d=100)}
    return data
```

Ці функції використовують трапецієподібну функцію належності для визначення ступеня належності значення до категорій "low", "normal" і "high" для систолічного тиску, діастолічного тиску, пульсу та загальної оцінки артеріального тиску.

Визначення правил нечіткої системи

```
# Визначення правил нечіткої системи
def calculate_rules(systolic, diastolic, pulse):
    return [
        # Правило 1: Якщо (систолічний = high і діастолічний = high) або пульс = high, то оцінка = high
        ('high', max(min(systolic['high'], diastolic['high']), pulse['high'])),

        # Правило 2: Якщо систолічний = normal і діастолічний = normal і пульс = normal, то оцінка = normal
        ('normal', min(systolic['normal'], diastolic['normal'], pulse['normal'])),

        # Правило 3: Якщо (систолічний = low і діастолічний = low) або пульс = low, то оцінка = low.
        ('low', max(min(systolic['low'], diastolic['low']), pulse['low'])),

        # Правило 4: Якщо систолічний = normal і діастолічний = normal і пульс = high, то оцінка = normal
        ('normal', min(systolic['normal'], diastolic['normal'], pulse['high'])),
    ]
```

Ця функція визначає правила нечіткої логіки для оцінки артеріального тиску на основі фазифікованих значень систолічного тиску, діастолічного тиску та пульсу.

Імплікація та агрегація правил

```
# Імплікації відповідних правил (Мамдані)
def calculate_implication(rules):
    num = 100
    blood_pressure_values = np.linspace(start=0, stop=100, num=num)
    blood_pressure_fuzzy = {'low': np.zeros(num), 'normal': np.zeros(num), 'high':
np.zeros(num)}

    for i in range(num):
        fuzzy_output = blood_pressure_func(value=blood_pressure_values[i])
        blood_pressure_fuzzy['low'][i] = fuzzy_output['low']
        blood_pressure_fuzzy['normal'][i] = fuzzy_output['normal']
        blood_pressure_fuzzy['high'][i] = fuzzy_output['high']

    result = {'low': np.zeros(num), 'normal': np.zeros(num), 'high': np.zeros(num)}
    counter = {'low': 0, 'normal': 0, 'high': 0}
    for name, value in rules:
        result[name] += np.minimum(value, blood_pressure_fuzzy[name])
        counter[name] += 1

    for key in result.keys():
        result[key] /= counter[key]
    return result
```

Ця функція обчислює імплікацію правил нечіткої логіки за Мамдані, створюючи нечіткі множини для значень артеріального тиску та обчислюючи результати імплікації.

Агрегація результатів імплікації

```
# Агрегації результатів імплікації правил
def calculate_aggregation(implication):
    result = np.zeros_like(implication[list(implication.keys())[0]])
    for name, value in implication.items():
        result = np.maximum(result, value)
    return result
```

Ця функція об'єднує результати імплікації, обираючи максимальне значення для кожного значення артеріального тиску.

Дефазифікація результатів

```
# Проведення дефазифікації результатів
def calculate_defuzzification(aggregation):
    num = len(aggregation)
    blood_pressure_values = np.linspace(start=0, stop=100, num=num)

    a, b = 0, 0
    for i in range(num):
        if aggregation[i] > 0:
            a += aggregation[i] * blood_pressure_values[i]
            b += aggregation[i]
    return a / b if not b == 0 else None
```

Ця функція проводить дефазифікацію, обчислюючи центроїд нечіткої множини для визначення остаточного значення артеріального тиску.

Функція для обчислення оцінки артеріального тиску

```
# Функція для обчислення оцінки артеріального тиску
def calculate_blood_pressure(systolic_value, diastolic_value, pulse_value):
    # Проведення фазифікації вхідних параметрів
    systolic_fuzzy = systolic_func(value=systolic_value)
    diastolic_fuzzy = diastolic_func(value=diastolic_value)
    pulse_fuzzy = pulse_func(value=pulse_value)

    # Обчислення відповідних правил
    rules = calculate_rules(systolic=systolic_fuzzy, diastolic=diastolic_fuzzy,
pulse=pulse_fuzzy)

    # Обчислення імплікації за Мамдані
    implication = calculate_implication(rules=rules)

    # Проведення агрегації вілповідних результатів
    aggregation = calculate_aggregation(implication=implication)

    # Дефазивікація ортиманої оцінки кровяного тиску
    blood_pressure_score = calculate_defuzzification(aggregation=aggregation)

    if blood_pressure_score is None:
        return (100 - 0) / 2
    return blood_pressure_score
```

Ця функція об'єднує всі попередні етапи для обчислення оцінки артеріального тиску на основі вхідних значень систолічного тиску, діастолічного тиску та пульсу.

Графічний інтерфейс користувача

```
def interface():
    try:
        systolic_value = float(entry_systolic.get())
        diastolic_value = float(entry_diastolic.get())
        pulse_value = float(entry_pulse.get())

        if not (80 <= systolic_value <= 180):
            messagebox.showerror('Помилка', 'Межі систолічного тиску: 80 - 180')

        if not (50 <= diastolic_value <= 120):
            messagebox.showerror('Помилка', 'Межі діастолічного тиску: 50 - 120')

        if not (40 <= pulse_value <= 180):
            messagebox.showerror('Помилка', 'Межі пульсу: 40 - 180')

        result = calculate_blood_pressure(
            systolic_value=systolic_value,
            diastolic_value=diastolic_value,
            pulse_value=pulse_value
        )
        messagebox.showinfo('Результат', f'Оцінка артеріального тиску: {result:.2f}')
    except ValueError:
        messagebox.showerror('Помилка', 'Будь ласка, введіть числові значення для
всіх полів.')
```

Ця функція реалізує логіку взаємодії користувача з графічним інтерфейсом. Вона зчитує вхідні значення, перевіряє їх на коректність та обчислює оцінку артеріального тиску.

Головна функція (точка входу)

```
if __name__ == '__main__':
    # Створення графічного інтерфейсу користувача
    root = tk.Tk()
    root.title('Оцінка артеріального тиску')

    tk.Label(root, text='Систолічний тиск').grid(row=0, column=0)
```

```
    entry_systolic = tk.Entry(root)
    entry_systolic.grid(row=0, column=1)

    tk.Label(root, text='Діастолічний тиск').grid(row=1, column=0)
    entry_diastolic = tk.Entry(root)
    entry_diastolic.grid(row=1, column=1)

    tk.Label(root, text='Пульс').grid(row=2, column=0)
    entry_pulse = tk.Entry(root)
    entry_pulse.grid(row=2, column=1)

    tk.Button(root, text='Обчислити', command=interface).grid(row=3, column=0,
columnspan=2)
    root.mainloop()
```

Цей блок коду створює графічний інтерфейс, розміщуючи відповідні елементи (поля введення та кнопки) та запускає головний цикл програми.

test.py

Необхідні бібліотеки

```
import numpy as np
from matplotlib import pyplot as plt
from main import systolic_func, diastolic_func, pulse_func, calculate_blood_pressure
```

`numpy` -- для числових обчислень

`matplotlib` -- для створення графічного інтерфейсу

`main` -- це власний файл (main.py), це для отримання функції з нього

Встановлення початкових умов

```
np.random.seed(seed=2024)
USE_NOISE = 1  # 1 = використання шуму, 0 = відсутність шуму
```

Цей блок встановлює початкове значення для генератора випадкових чисел, щоб результати були відтворюваними, та визначає, чи буде використовуватися шум у даних.

Генерація даних

```
# Визначення часового проміжку
num = 100
time = np.linspace(start=0, stop=1, num=num)

# Зміна систолічного тиску з часом + шум
systolic = (180 - 80) * 0.8 * time + 90 + 3 * np.random.randn(num) * USE_NOISE

# Зміна діастолічного тиску з часом + шум
```

```
diastolic = (120 - 50) * 0.8 * time + 60 + 2 * np.random.randn(num) * USE_NOISE

# Зміна пульсу з часом + шум
pulse = (180 - 40) * 0.8 * time + 50 + 3 * np.random.randn(num) * USE_NOISE
```

Цей блок коду генерує дані для систолічного тиску, діастолічного тиску та пульсу. Значення змінюються з часом та включають шум, який моделюється випадковими числами з нормального розподілу, помноженими на `USE_NOISE`.

Обчислення оцінки артеріального тиску

```
# Обчислення оцінки артеріального тиску у часі
blood_pressure = [calculate_blood_pressure(systolic_value=systolic[t],
                                           diastolic_value=diastolic[t],
                                           pulse_value=pulse[t]) for t in range(num)]
```

Цей блок обчислює оцінку артеріального тиску для кожного моменту часу, використовуючи функцію `calculate_blood_pressure` та згенеровані дані для систолічного тиску, діастолічного тиску та пульсу.

Відображення графіку

```
# Відображення графіку
plt.figure(figsize=(10, 8))

plt.subplot(5, 1, 1)
plt.plot(systolic, color="tab:blue")
plt.ylabel('Систолічний')
plt.xticks([])

plt.subplot(5, 1, 2)
plt.plot(diastolic, color="tab:orange")
plt.ylabel('Діастолічний')
plt.xticks([])

plt.subplot(5, 1, 3)
plt.plot(pulse, color="tab:green")
plt.ylabel('Пульс')
plt.xticks([])
```

```
plt.subplot(5, 1, (4, 5))
plt.plot(blood_pressure, color="tab:purple")
plt.title('Оцінка артеріального тиску')
plt.xticks([])

plt.tight_layout()
plt.show()
```

Цей блок коду створює графік, що складається з чотирьох підграфіків:

1. Систолічний тиск за часом.
2. Діастолічний тиск за часом.
3. Пульс за часом.
4. Оцінка артеріального тиску за часом.

Використовуючи функцію plt.subplot, кожен підграфік розташовується один під одним. А plt.show() відображає графік.

```
# Plotting the membership functions for systolic, diastolic, and pulse as in slide 4
x_systolic = np.linspace(60, 200, 400)
x_diastolic = np.linspace(40, 140, 400)
x_pulse = np.linspace(30, 190, 400)

systolic_low = [systolic_func(value)['low'] for value in x_systolic]
systolic_normal = [systolic_func(value)['normal'] for value in x_systolic]
systolic_high = [systolic_func(value)['high'] for value in x_systolic]

diastolic_low = [diastolic_func(value)['low'] for value in x_diastolic]
diastolic_normal = [diastolic_func(value)['normal'] for value in x_diastolic]
diastolic_high = [diastolic_func(value)['high'] for value in x_diastolic]

pulse_low = [pulse_func(value)['low'] for value in x_pulse]
pulse_normal = [pulse_func(value)['normal'] for value in x_pulse]
pulse_high = [pulse_func(value)['high'] for value in x_pulse]
```

В цьому блоці йде генерація значень для показу значень в функціях належності.

```
plt.figure(figsize=(15, 10))

plt.subplot(3, 1, 1)
plt.plot(x_systolic, systolic_low, label='Low', color='tab:blue')
plt.plot(x_systolic, systolic_normal, label='Normal', color='tab:green')
plt.plot(x_systolic, systolic_high, label='High', color='tab:red')
plt.axhline(0, color='black', linewidth=3)
plt.title('Систолічний тиск')
plt.xlabel('Значення')
plt.ylabel('Належність')
```

```
plt.legend()
plt.xlim(80, 180)

plt.subplot(3, 1, 2)
plt.plot(x_diastolic, diastolic_low, label='Low', color='tab:blue')
plt.plot(x_diastolic, diastolic_normal, label='Normal', color='tab:green')
plt.plot(x_diastolic, diastolic_high, label='High', color='tab:red')
plt.axhline(0, color='black', linewidth=3)
plt.title('Діастолічний тиск')
plt.xlabel('Значення')
plt.ylabel('Належність')
plt.legend()
plt.xlim(50, 120)

plt.subplot(3, 1, 3)
plt.plot(x_pulse, pulse_low, label='Low', color='tab:blue')
plt.plot(x_pulse, pulse_normal, label='Normal', color='tab:green')
plt.plot(x_pulse, pulse_high, label='High', color='tab:red')
plt.axhline(0, color='black', linewidth=3)
plt.title('Пульс')
plt.xlabel('Значення')
plt.ylabel('Належність')
plt.legend()
plt.xlim(40, 180)

plt.tight_layout()
plt.show()
```

Цей блок коду створює графіки функцій належності для кожного нашого параметру.