Лабораторная работа № 1

«Методы разработки графических пользовательских интерфейсов»

*Цель работы*: приобретение практических навыков по созданию графический интерфейсов пользователя с помощью Модуля tkinter.

Задание к лабораторной работе

Разработать программное приложение с графическим интерфейсом, согласно теме индивидуального варианта (Приложение 1).

Ход работы:

Задание 1. Вычислить сумму положительных элементов нижней треугольной матрицы.

import tkinter as tk

def calculate\_sum(matrix):

    # Функция для вычисления суммы положительных элементов в нижнем треугольнике матрицы

    total\_sum = 0

    n = len(matrix)

    for i in range(n):

        for j in range(i + 1):

            if matrix[i][j] > 0:

                total\_sum += matrix[i][j]

    return total\_sum

def on\_calculate():

    # Обработчик события кнопки "Вычислить"

    matrix = [

        [int(entry.get()) for entry in row] for row in entries

    ]

    result = calculate\_sum(matrix)

    result\_label.config(text=f"Сумма положительных элементов: {result}")

# Создание основного окна

root = tk.Tk()

root.title("Вычисление суммы положительных элементов")

# Создание таблицы ввода элементов матрицы

entries = []

for i in range(3):  # Замени на нужный размер матрицы

    row\_entries = []

    for j in range(i + 1):

        entry = tk.Entry(root, width=5)

        entry.grid(row=i, column=j, padx=5, pady=5)

        row\_entries.append(entry)

    entries.append(row\_entries)

# Кнопка для вычисления суммы

calculate\_button = tk.Button(root, text="Вычислить", command=on\_calculate)

calculate\_button.grid(row=len(entries), columnspan=len(entries[-1]) + 1, pady=10)

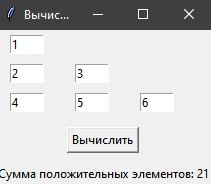
# Метка для вывода результата

result\_label = tk.Label(root, text="")

result\_label.grid(row=len(entries) + 1, columnspan=len(entries[-1]) + 1)

# Запуск главного цикла

root.mainloop()



Задание 2. Найти наименьший элемент матрицы и номера строки и столбца, в которых он расположен.

import tkinter as tk

def find\_minimum(matrix):

    # Функция для поиска наименьшего элемента и его позиции в матрице

    min\_element = float('inf')

    min\_row, min\_col = -1, -1

    n = len(matrix)

    for i in range(n):

        for j in range(len(matrix[i])):

            if matrix[i][j] < min\_element:

                min\_element = matrix[i][j]

                min\_row, min\_col = i, j

    return min\_element, min\_row, min\_col

def on\_find\_minimum():

    # Обработчик события кнопки "Найти минимум"

    matrix = [

        [int(entry.get()) for entry in row] for row in entries

    ]

    min\_element, min\_row, min\_col = find\_minimum(matrix)

    result\_label.config(text=f"Минимальный элемент: {min\_element}\n"

                             f"Строка: {min\_row + 1}, Столбец: {min\_col + 1}")

# Создание основного окна

root = tk.Tk()

root.title("Поиск минимального элемента")

# Создание таблицы ввода элементов матрицы

entries = []

for i in range(3):  # Замени на нужный размер матрицы

    row\_entries = []

    for j in range(3):  # Замени на нужный размер матрицы

        entry = tk.Entry(root, width=5)

        entry.grid(row=i, column=j, padx=5, pady=5)

        row\_entries.append(entry)

    entries.append(row\_entries)

# Кнопка для поиска минимума

find\_minimum\_button = tk.Button(root, text="Найти минимум", command=on\_find\_minimum)

find\_minimum\_button.grid(row=len(entries), columnspan=len(entries[-1]) + 1, pady=10)

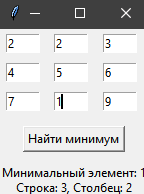
# Метка для вывода результата

result\_label = tk.Label(root, text="")

result\_label.grid(row=len(entries) + 1, columnspan=len(entries[-1]) + 1)

# Запуск главного цикла

root.mainloop()



Задание 3. Транспонировать матрицу, результат транспонирования вывести в форму таблицы.

import tkinter as tk

def transpose\_matrix(matrix):

    # Функция для транспонирования матрицы

    transposed = [[row[i] for row in matrix] for i in range(len(matrix[0]))]

    return transposed

def on\_transpose():

    # Обработчик события кнопки "Транспонировать"

    original\_matrix = [

        [int(entry.get()) for entry in row] for row in entries

    ]

    transposed\_matrix = transpose\_matrix(original\_matrix)

    # Очищаем предыдущие элементы таблицы, если они есть

    for row in result\_entries:

        for entry in row:

            entry.destroy()

    # Создаем новые элементы таблицы для транспонированной матрицы

    for i in range(len(transposed\_matrix)):

        row\_entries = []

        for j in range(len(transposed\_matrix[i])):

            entry = tk.Entry(root, width=5)

            entry.grid(row=i, column=j + len(original\_matrix[0]) + 1, padx=5, pady=5)

            entry.insert(0, str(transposed\_matrix[i][j]))

            row\_entries.append(entry)

        result\_entries.append(row\_entries)

# Создание основного окна

root = tk.Tk()

root.title("Транспонирование матрицы")

# Создание таблицы ввода элементов матрицы

entries = []

for i in range(3):  # Замени на нужный размер матрицы

    row\_entries = []

    for j in range(3):  # Замени на нужный размер матрицы

        entry = tk.Entry(root, width=5)

        entry.grid(row=i, column=j, padx=5, pady=5)

        row\_entries.append(entry)

    entries.append(row\_entries)

# Кнопка для транспонирования

transpose\_button = tk.Button(root, text="Транспонировать", command=on\_transpose)

transpose\_button.grid(row=len(entries), columnspan=len(entries[-1]) + 1, pady=10)

# Переменная для хранения элементов таблицы результата

result\_entries = []

# Запуск главного цикла

root.mainloop()

