

# ГРАФИЧЕСКИЙ ИНТЕРФЕЙС ПОЛЬЗОВАТЕЛЯ В PYTHON

Цель: *Создание графического интерфейса программы в Python.*

## Задание

1. Написать программу для решения задачи, в соответствии с вариантом в [Приложении 1](#).
  - 1.1. Исходные данные задачи и результаты расчета (кроме графиков) должны вводиться и выводиться посредством графического интерфейса, созданного при помощи библиотеки tkinter.
  - 1.2. Дизайн графического интерфейса должен быть авторским (запрещается использовать дизайн других студентов).
  - 1.3. Предусмотреть для всех полей для ввода значения по умолчанию (для быстрой проверки работоспособности программы).
  - 1.4. Для решения задач можно использовать любые функции библиотек numpy и scipy.
2. Создать отчет по проделанной работе. Отчет должен содержать:
  - 2.1. Титульный лист, цель работы, задание. Все страницы, кроме первой, должны быть пронумерованы в верхнем правом углу. В верхнем колонтитуле разместить с выравниванием влево фамилию и инициалы студента и надпись “Лабораторная работа №5”.
  - 2.2. Постановку каждой задачи.
  - 2.3. Математическую модель.
  - 2.4. Созданные блок-схемы.
  - 2.5. Листинг написанных программ.
  - 2.6. Таблицы тестирования.
  - 2.7. Текстовые пояснения, объясняющие ход выполнения работы по всем этапам.
  - 2.8. Присвоить файлу имя “Лабораторная работа №5”. Сохранить файл в соответствующей папке Google Диска. Оповестить преподавателя через почту о созданном отчете и прикрепить отчет.
  - 2.9. Получить замечания по документу, в соответствии с замечаниями внести изменения в документ. Оповестить преподавателя ответом на его письмо.

## Введение

Для взаимодействия с пользователем современные приложения используют оконный графический интерфейс. Его применение более предпочтительно, чем, например, использование командной строки, поскольку более наглядно и интуитивно понятно. GUI (graphic user interface, графический интерфейс пользователя) имеет в своем наборе удобные элементы управления. Например, выпадающие списки, радио-кнопки и т.п. Пример программы, использующий элементы графического интерфейса, приведен на рисунке 1.

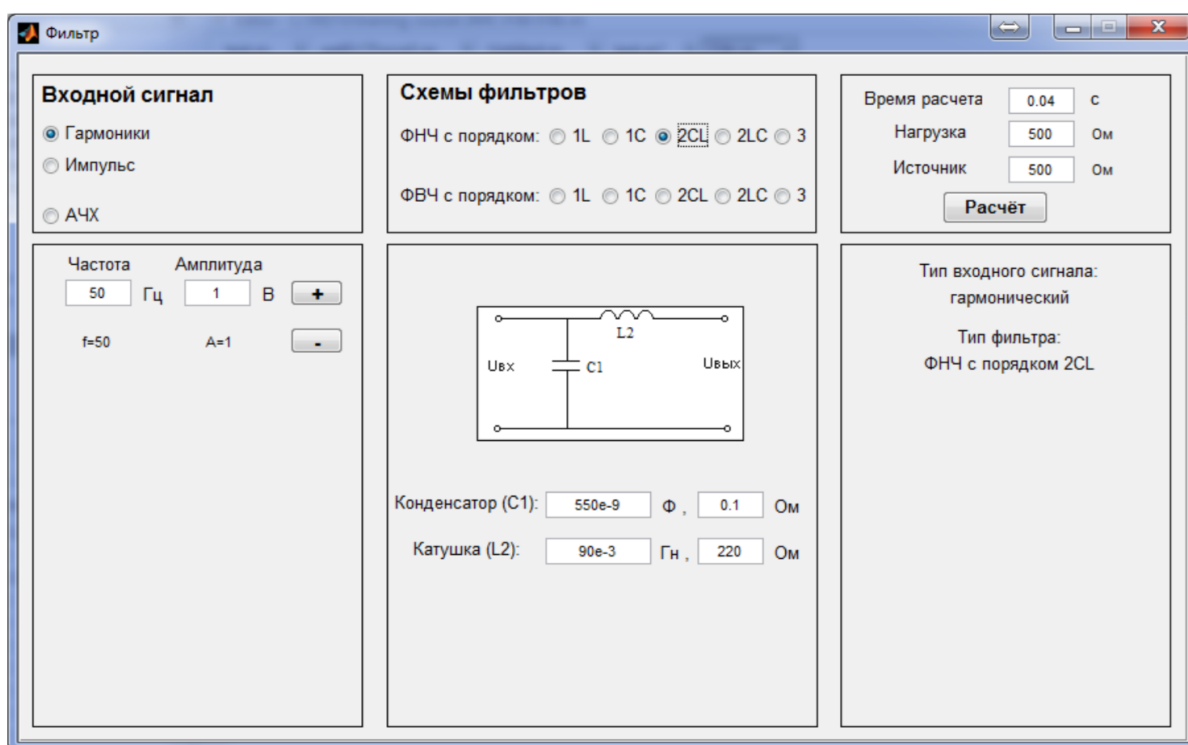


Рисунок 1 - Графический интерфейс программы для расчета фильтра

Python имеет множество библиотек для создания графического интерфейса, т.н. GUI-фреймворков. Наиболее известные из них: PyQt, tkinter, Kivy, PySide, dearpygui и т.д. Большинство GUI-фреймворков имеют открытый исходный код и кроссплатформенную реализацию, что делает их легкодоступными, гибкими и мощными инструментами для создания приложений с графическим интерфейсом.

Для выполнения этой лабораторной работы будет использоваться библиотека tkinter. Tkinter представляет интерфейс в Python для графической библиотеки Tk (название "Tkinter" является сокращением "Tk interface"). На сегодняшний день и библиотека Tk, и сам тулkit tkinter доступны для большинства операционных систем, в том числе для Mac OS, Linux и Windows.

Преимущества Tkinter:

- Данный тулkit по умолчанию включен в стандартную библиотеку языка Python в виде отдельного модуля, поэтому не потребуется что-то дополнительно устанавливать;
- Tkinter - кроссплатформенный, один и тот же код будет работать одинаково на разных платформах (Mac OS, Linux и Windows);
- Tkinter легко изучать. Сам тулkit, хотя и содержит некоторый готовый код, виджеты и графические элементы, но при этом довольно лаконичен и прост.
- Tk распространяется по BSD-лицензии, поэтому библиотека может быть использована как в опенсорсных проектах, так и в коммерческих разработках.

Много информации о библиотеке tkinter есть на сайте:  
<https://metanit.com/python/tkinter/>

## Основное окно приложения

Основным графическим объектом является окно. Окно задается с помощью функции Tk. Функция возвращает ссылку на объект Tk (основное окно интерфейса):

```
# Пробная программа с графическим интерфейсом  
from tkinter import *  
my_window = Tk()
```

Здесь: `my_window` - это переменная (объект) класса Tk, -указывающая на окно. Используется для дальнейшего обращения к окну с целью задания/изменения его свойств. Результатом работы приведенного выше кода будет окно, параметры которого будут заданы по умолчанию (рис.2).



Рисунок 2 - Внешний вид окна с параметрами по умолчанию

**Важно!** Может быть создано только одно окно класса Tk (его называют корневым). Если закрыть это окно, то выполнение программы прекращается. Если необходимо создать графический интерфейс, состоящий из нескольких окон, все последующие окна, кроме корневого, создаются при помощи класса `toplevel` (создание многооконных интерфейсов будет рассмотрено в следующей лабораторной работе).

Класс Tk обладает набором методов, позволяющих изменить свойства окна.

- Размеры и положение окна. Задаются при помощи метода `geometry`. Аргументом функции является текстовая строка, содержащая размеры окна в формате “320x200” (ширина и высота окна в пикселях), или в формате “320x200+400+300” (ширина x высота + координата x левого нижнего угла окна относительно левого нижнего угла экрана + координата y левого нижнего угла окна относительно левого нижнего угла экрана). Например, приведенный ниже код позволяет создать окно размером 600 на 400 пикселей, расположенное строго в центре экрана (разрешение экрана 1900x1080 пикселей)

```
# Разрешение экрана в пикселях
we = 1920; he = 1080
# Размеры окна в пикселях
ww = 600; hw = 400
# Управляющая строка
s = (str(ww)+"x"+str(hw)+"+"
     + str(int(we/2-ww/2))+ "+"
     + str(int(he/2-hw/2)))
# Задание размеров
my_window.geometry(s)
```

- Метод `title` позволяет установить заголовок окна (по умолчанию там надпись `tk`):

```
my_window.title('Моя программа')
```

- Метод `iconbitmap` позволяет установить иконку (по умолчанию там символ пера). В функцию `iconbitmap` передаётся имя файла с иконкой (файл должен быть в формате \*.ico)

```
my_window.iconbitmap(default = 'some_icon.ico')
```

- Метод `resizable` позволяет отключить пользователю возможность

растягивать окно. Первый аргумент (значения True/False) - позволяет/запрещает растягивать окно по горизонтали, второй - по вертикали.

```
my_window.resizable(False,False)
```

- Большое количество свойств окна не имеет специального метода для редактирования. Чтобы изменить их используется общий метод `configure`, в котором указывается название свойства и его значение. В частности, для изменения цвета фона окна можно задать свойство `bg`:

```
my_window.configure(bg = '#28B2D1')
```

Здесь `#28B2D1` - это шестнадцатеричный код цвета (интенсивности красного, зеленого и синего цветов в диапазоне от 0 до 255, записанные в шестнадцатеричной системе счисления).

В результате применения приведенных выше функций, получим окно, приведенное на рис.3.

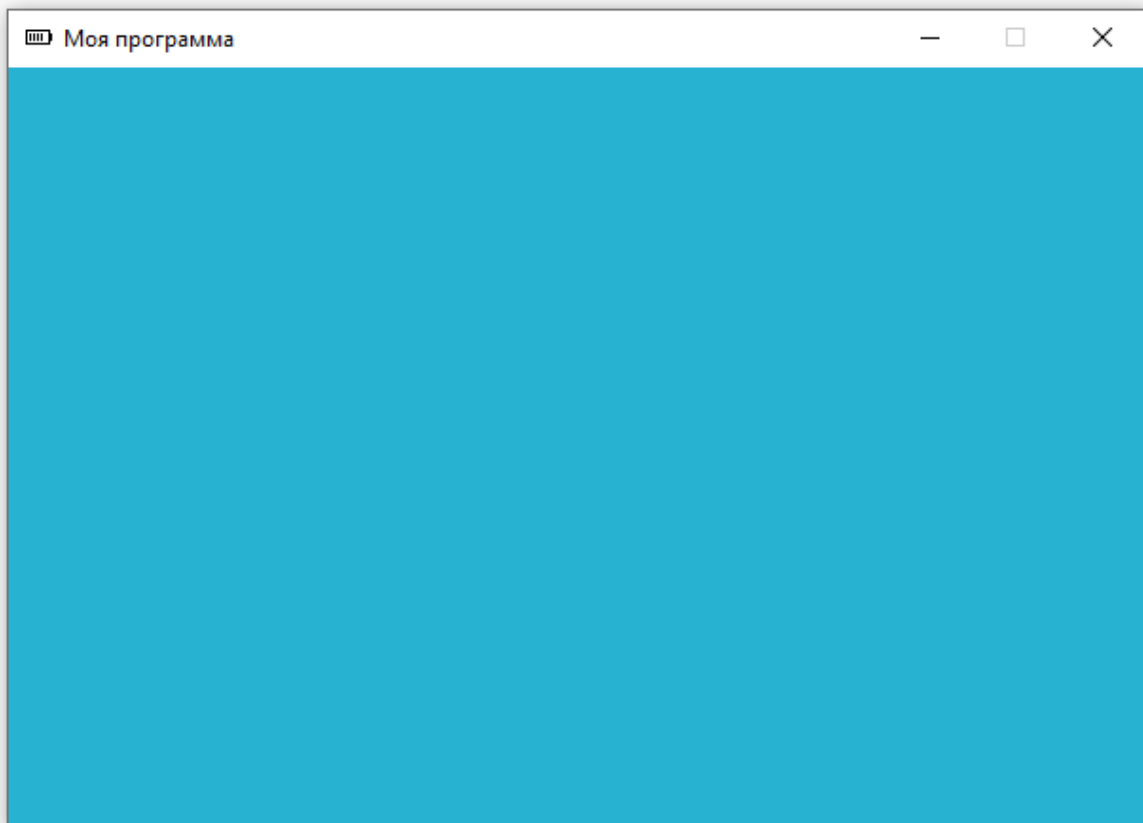


Рисунок 3 - Результат изменения свойств окна

## Виджеты. Создание текстовых меток

Элементы графического интерфейса (текстовые метки, кнопки, поля ввода и др.) библиотеки tkinter называют виджетами. Виджеты находятся в отдельной библиотеке tkinter.ttk. Импортировать её не нужно, если в начале программы был использован код (этот код импортирует все функции библиотеки tkinter):

```
from tkinter import *
```

Самый базовый виджет - текстовая метка. Этот виджет позволяет выводит на интерфейс статический текст без возможности редактирования. Для создания текстовой метки применяется конструктор, общий вид которого следующий:

```
Label(master, options)
```

где: master - ссылка на родительский объект (основное графическое окно); options - свойства виджета. Свойства виджета задаются в виде операторов присваивания: имя\_свойства = значение свойства, перечисляемых через запятую.

После создания, виджет должен быть размещен на родительском объекте. Для этого используется метод place. Можно использовать метод place с абсолютными координатами (в пикселях). Координаты отсчитываются от левого верхнего угла основного окна до левого верхнего угла виджета. Ниже приведен код, создающий текстовые метки на ранее сгенерированном основном окне (предполагается написать программу, решающую квадратное уравнение).

```
# Создание текстовых меток
```

```
S1 = Label(  
    my_window,  
    text='Решение квадратного уравнения',  
    background = '#28B2D1',  
    foreground = '#FFFFFF',  
    font=('Times New Roman',28))  
S1.place(x = 40,y = 0)  
S2 = Label(  
    my_window,  
    text='A = ',  
    background = '#28B2D1',  
    foreground = '#FFFFFF',  
    font=('Times New Roman',14))  
S2.place(x = 60, y = 200)  
S3 = Label(  
    my_window, text='B = ',  
    background = '#28B2D1',  
    foreground = '#FFFFFF',  
    font=('Times New Roman',14))  
S3.place(x = 240, y = 200)
```

```
S4 = Label(
    my_window, text='C = ',
    background = '#28B2D1',
    foreground = '#FFFFFF',
    font=('Times New Roman',14))
S4.place(x = 420, y = 200)
```

В таблице 1 приведены некоторые основные свойства текстовых меток.

Таблица 1 - Основные свойства текстовых меток

Название свойства	Что задает и возможные значения
text	Текст метки. Тип данных - строка
background	Цвет фона. Тип данных - строка (можно использовать шестнадцатеричный код RGB цвета после символа решетки).
foreground	Цвет текста. Тип данных - строка (можно использовать шестнадцатеричный код RGB цвета после символа решетки).
font	Шрифт текста и его размер. Формат свойства: font = (название_шрифта_строка, размер_шрифта_число)
justify	Выравнивание текста. LEFT - по левому краю, CENTER - по центру, RIGHT - по правому краю

Любопытно, но в библиотеке tkinter текстовые метки также используются для вывода на графическое окно изображений. Для этого используется свойство image. Загрузить изображение можно при помощи функции PhotoImage (файл с изображением должен лежать в той же папке, где и программа python).

```
# Загрузка изображения
loaded_image1 = PhotoImage(file = 'equation.png')
S5 = Label(image = loaded_image1)
S5.place(x = 50, y = 60)
```

Результат создания текстовых меток и загрузки изображения приведен на рис.4.

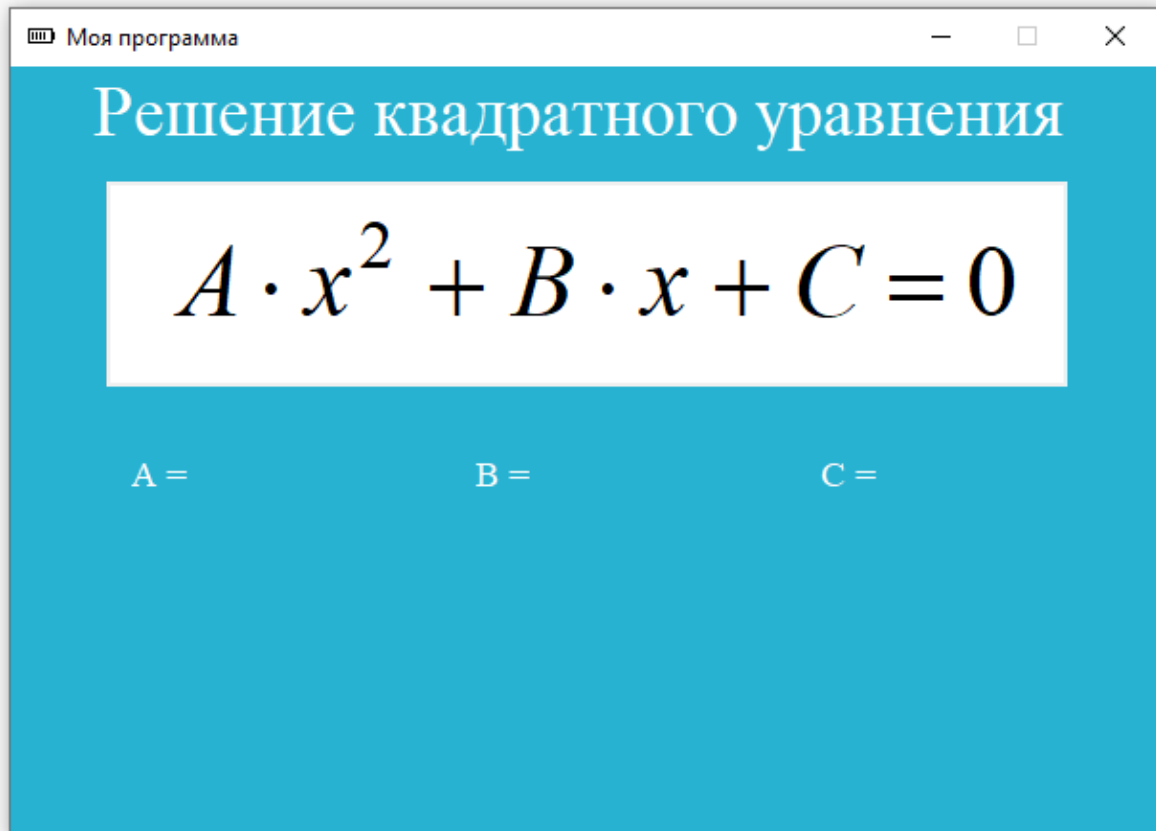


Рисунок 4 - Результат создания текстовых меток и загрузки изображения

## Виджеты. Создание полей ввода

Поля ввода создаются при помощи конструктора Entry. Использование этого конструктора - аналогично конструктору Label. Многие свойства виджета Entry аналогичны свойствам виджета Label (табл.1). Приведенный ниже код создает поля для ввода коэффициентов A,B,C квадратного уравнения.

```
# Создание полей ввода
E1 = Entry(
    my_window,
    font = ('Times New Roman',14),
    justify = RIGHT)
E1.place(x = 100, y = 200, width = 80)
E2 = Entry(
    my_window,
    font = ('Times New Roman',14),
    justify = RIGHT)
E2.place(x = 280, y = 200, width = 80)
E3 = Entry(
```



```
my_window,  
font = ('Times New Roman',14),  
justify = RIGHT)  
E3.place(x = 460, y = 200, width = 80)
```

По умолчанию поля ввода создаются пустыми. Чтобы задать текст, находящийся в них по умолчанию, можно использовать метод insert:

```
# Задание текста по умолчанию  
E1.insert(0,'1.0')  
E2.insert(0,'-3.0')  
E3.insert(0,'2.0')
```

Считать значение поля можно при помощи метода get:

```
print('Значение поля A = ', E1.get())
```

## Виджеты. Привязка к переменным.

Виджет Label используется для создания статического текста. Изменить его после создания затруднительно. Но иногда требуется иметь возможность изменить текст созданного графического объекта, например, для вывода результатов расчета. Для этого используется привязка виджета к специальным переменным. В tkinter используется четыре типа специальных переменных:

- StringVar - строковая переменная;
- IntVar - целочисленная переменная;
- BooleanVar - логическая переменная;
- DoubleVar - вещественная переменная.

При изменении значения специальной переменной, автоматически перерисовывается связанный с ней виджет. И наоборот - изменение виджета (например, ввод текста в поле для ввода) автоматически изменит связанную с ним специальную переменную.

Для установления значения специальной переменной при её создании используется свойство value. Приведенный ниже код создает строковые переменные для вывода результатов решения.

```
# Создание строковых переменных для вывода  
out1 = StringVar()  
out2 = StringVar()
```

Для связи созданных строковых переменных с виджетом Label используется свойство textvariable. Следующий код создает две текстовых метки, привязанные к переменным out1 и out2.

```
# Создание текстовых полей для вывода результата
S6 = Label(
    my_window,
    font = ('Times New Roman',14),
    justify = LEFT,
    background = '#28B2D1',
    foreground = '#FFFFFF',
    textvariable = out1)
S6.place(x = 60, y = 270)
S7 = Label(
    my_window,
    font = ('Times New Roman',14),
    justify = RIGHT,
    background = '#28B2D1',
    foreground = '#FFFFFF',
    textvariable = out2)
S7.place(x = 60, y = 320)
```

Установить значение специальной переменной можно при помощи метода set:

```
out1.set('X1 = ' + str(x1))
```

## Виджеты. Создание кнопок. Основной цикл

Кнопки являются одними из наиболее важных виджетов. Они добавляют приложению с графическим интерфейсом возможность реагировать на действия пользователя. Кнопки создаются при помощи конструктора Button. Основным свойством кнопки является свойство command. Этому свойству необходимо присвоить имя функции, которая будет вызываться при нажатии на кнопку.

Напишем функцию, которая выполняет решение квадратного уравнения. Общая концепция работы данной функции: сначала считываются данные, введенные в объекты Entry. Затем выполняется расчет. Результаты расчета вводятся в текстовые метки, приготовленные для вывода результата (путем изменения связанных с ними специальных переменных StringVar).

```

# Функция, вызываемая при нажатии кнопки
def on_click_button1():
    global E1, E2, E3, out1, out2
    # Считывание данных с полей ввода
    # Данные считываются в виде строк
    # Нужно преобразовать в вещественный тип
    A = float(E1.get())
    B = float(E2.get())
    C = float(E3.get())
    # Решение уравнения и вывод результата
    if A==0 and B==0 and not C==0:
        out1.set('Неправильное уравнение')
        out2.set('')
        return
    if A==0 and B==0 and C==0:
        out1.set('X - любое число')
        out2.set('')
        return
    if A==0:
        x = -C/B
        out1.set('Линейное уравнение')
        out2.set('X = ' + str(x))
        return
    D = B**2 - 4*A*C
    if D<0:
        out1.set('Нет действительных решений')
        out2.set('')
        return
    x1 = (-B + np.sqrt(D))/2/A
    x2 = (-B - np.sqrt(D))/2/A
    out1.set('X1 = ' + str(x1))
    out2.set('X2 = ' + str(x2))

```

Приведенный ниже код создает кнопку, нажатие на которую вызовет функцию on\_click\_button1.

```

# Создание кнопки
btn1 = Button(
    my_window,

```

```
text = 'Решить уравнение',  
command = on_click_button1)  
btn1.place(x = 360, y = 280, width = 200, height = 50)
```

После того, как графический интерфейс пользователя создан, необходимо запустить основной цикл работы программы. В этом режиме программа будет реагировать на события (клики мышью по кнопкам, ввод данных в поля и т.д.). Запуск основного цикла выполняет метод `mainloop` главного окна:

```
# Запуск основного цикла  
my_window.mainloop()
```

Итоговый вид графического интерфейса программы для решения квадратного уравнения и результат нажатия на кнопку “решить уравнение” приведены на рис.5.

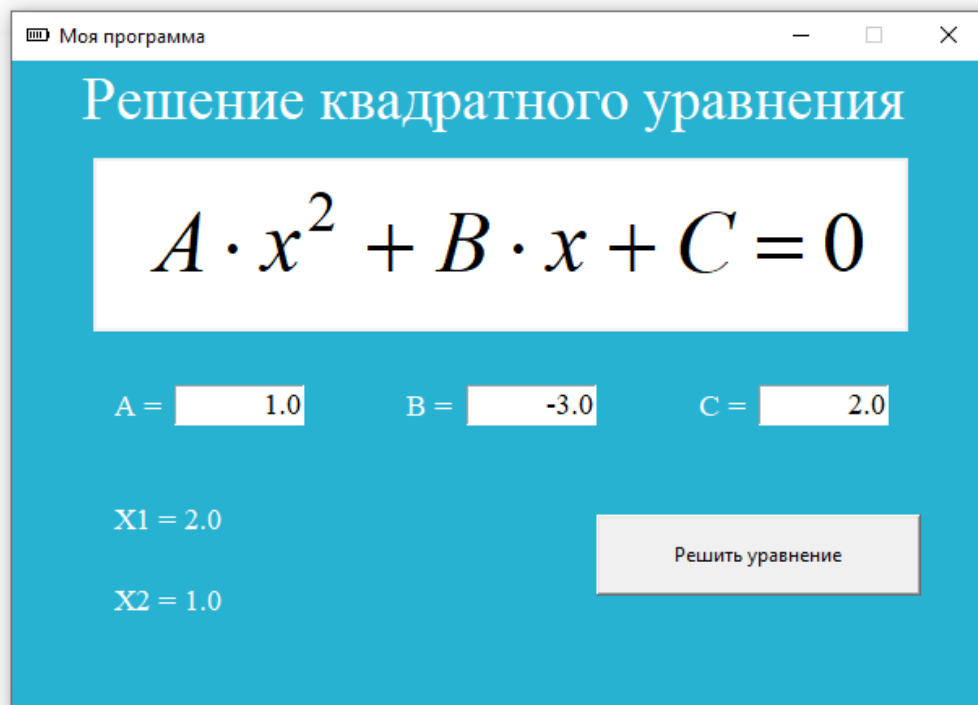


Рисунок 5 - Итоговый вид графического интерфейса

## Методические указания

Рассмотрим создание графического интерфейса на примере задачи расчета тока в электрической цепи, приведенной на рисунке 6.

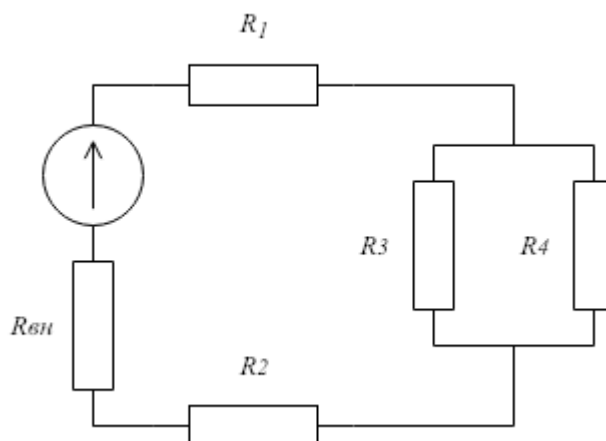


Рисунок 6 – Пример схемы сопротивлений

Для данной схемы (рисунок 6) составим уравнение для расчета эквивалентного сопротивления. Здесь сопротивления  $R_3$  и  $R_4$  соединены параллельно:

$$R_{34} = \frac{R_3 \cdot R_4}{R_3 + R_4}$$

Полученное сопротивление  $R_{34}$  соединено с сопротивлениями  $R_1$  и  $R_2$  последовательно. В итоге получим формулу для эквивалентного сопротивления:

$$R_{\text{экв}} = R_1 + R_2 + R_{34} = R_1 + R_2 + \frac{R_3 \cdot R_4}{R_3 + R_4}$$

Для расчета общего тока, потребляемого элементами схемы, воспользуемся законом Ома. Зависимость между напряжением  $E$  источника питания с внутренним сопротивлением  $R_{\text{вн}}$ , током электрической цепи и общим эквивалентным сопротивлением всей цепи определяется выражением:

$$I = \frac{E}{R_{\text{вн}} + R_{\text{экв}}}$$

Далее создаем графический интерфейс:

```
# Расчет тока в электрической цепи
from tkinter import *
```

```
my_window = Tk()
```

```
# Разрешение экрана в пикселях
```

```
we = 1920; he = 1080
```

```
# Размеры окна в пикселях
```

```
ww = 650; hw = 550
```

```

# Управляющая строка
s = (str(ww)+'x'+str(hw)+'+'
      + str(int(we/2-ww/2))+'+')
      + str(int(he/2-hw/2)))
# Задание размеров
my_window.geometry(s)
my_window.title('Расчет тока')
my_window.resizable(False,False)

```

Создаем текстовые метки.

```

# Текстовые метки
Label(my_window,
      text = 'Источник',
      font = ('Times New Roman',22)).\
      place(x = 50, y = 50)
Label(my_window,
      text = 'Сопротивления',
      font = ('Times New Roman',22)).\
      place(x = 400, y = 50)
Label(my_window,
      text = 'E = ',
      font = ('Times New Roman',18)).\
      place(x = 50, y = 120)
Label(my_window,
      text = 'B',
      font = ('Times New Roman',18)).\
      place(x = 200, y = 120)
Label(my_window,
      text = 'Rвн = ',
      font = ('Times New Roman',18)).\
      place(x = 50, y = 180)
Label(my_window,
      text = 'Om',
      font = ('Times New Roman',18)).\
      place(x = 200, y = 180)
Label(my_window,
      text = 'R1 = ',
      font = ('Times New Roman',18)).\
      place(x = 400, y = 120)

```

```

Label(my_window,
      text = 'Ом',
      font = ('Times New Roman',18)).\
      place(x = 550, y = 120)
Label(my_window,
      text = 'R2 = ',
      font = ('Times New Roman',18)).\
      place(x = 400, y = 180)
Label(my_window,
      text = 'Ом',
      font = ('Times New Roman',18)).\
      place(x = 550, y = 180)
Label(my_window,
      text = 'R3 = ',
      font = ('Times New Roman',18)).\
      place(x = 400, y = 240)
Label(my_window,
      text = 'Ом',
      font = ('Times New Roman',18)).\
      place(x = 550, y = 240)
Label(my_window,
      text = 'R4 = ',
      font = ('Times New Roman',18)).\
      place(x = 400, y = 300)
Label(my_window,
      text = 'Ом',
      font = ('Times New Roman',18)).\
      place(x = 550, y = 300)

```

Загружаем файл с изображением схемы и помещаем изображение на интерфейс.

```

# Загружаем и выводим изображение
loaded_image1 = PhotoImage(file = 'scheme1.png')
Label(image = loaded_image1).\
      place(x = 20, y = 240)

```

Создаем поля для ввода данных.

```

# Создаем поля для ввода данных
sE = Entry(my_window,
            justify = RIGHT,
            font = ('Times New Roman',18))
sE.place(x = 100, y = 120, width = 80)
sE.insert(0,'10.0')
sRvn = Entry(my_window,
              justify = RIGHT,
              font = ('Times New Roman',18))
sRvn.place(x = 100, y = 180, width = 80)
sRvn.insert(0,'2.0')
sR1 = Entry(my_window,
             justify = RIGHT,
             font = ('Times New Roman',18))
sR1.place(x = 460, y = 120, width = 80)
sR1.insert(0,'5.0')
sR2 = Entry(my_window,
             justify = RIGHT,
             font = ('Times New Roman',18))
sR2.place(x = 460, y = 180, width = 80)
sR2.insert(0,'10.0')
sR3 = Entry(my_window,
             justify = RIGHT,
             font = ('Times New Roman',18))
sR3.place(x = 460, y = 240, width = 80)
sR3.insert(0,'15.0')
sR4 = Entry(my_window,
             justify = RIGHT,
             font = ('Times New Roman',18))
sR4.place(x = 460, y = 300, width = 80)
sR4.insert(0,'20.0')

```

Создаем строковую переменную и текстовую метку для вывода результата.

```

# Строковая переменная и текстовая метка
# для вывода результата расчета
out1 = StringVar(value = 'I = ')
Label(my_window,
      textvariable = out1,

```



```
font = ('Times New Roman',18)).\
place(x = 400, y = 370)
```

Функция, вызываемая при нажатии кнопки и выполняющая расчет:

```
# Функция, вызываемая по кнопке
def on_click_button1():
    global sE, sRvn, sR1, sR2, sR3, sR4
    # Считываем данные с полей для ввода
    E = float(sE.get())
    Rvn = float(sRvn.get())
    R1 = float(sR1.get())
    R2 = float(sR2.get())
    R3 = float(sR3.get())
    R4 = float(sR4.get())
    # Рассчитываем ток
    R34 = 1/(1/R3 + 1/R4)
    Rekv = R1 + R2 + R34
    I = np.round(E / (Rvn + Rekv),2)
    # Выводим результат
    out1.set('I = ' + str(I) + ' A')
```

Создание кнопки:

```
# Создаем кнопку
Button(my_window,
    text = 'Рассчитать',
    command = on_click_button1).\
place(x = 400, y = 450, width = 200, height = 50)
```

Пример оконного интерфейса для расчета тока показан на рисунке 7.

Расчет тока

Источник

$E =$

10.0

В

$R_{вн} =$

2.0

Ом

Сопротивления

$R1 =$

5.0

Ом

$R2 =$

10.0

Ом

$R3 =$

15.0

Ом

$R4 =$

20.0

Ом

$I = 0.39$

А

Рассчитать

Рисунок 7 – Пример оконного интерфейса программы для расчета тока

После запуска программы следует убедиться, что в командном окне не появляются ошибки. Если ошибки имеются, следует их исправить. После этого необходимо провести тестирование данной программы (таблица 2) .

Таблица 2 - Таблица тестирования

Исходные значения	Результат программы	Верный результат	Сравнение
$E=1, R_{вн}=1, R1=1, R2=1, R3=1, R4=1$	$I=0.28571$ А	$I=0.286$ А	+
$E=10, R_{вн}=1, R1=1, R2=2, R3=3, R4=4$	$I=1.75$ А	$I=1.75$ А	+

## Контрольные задания

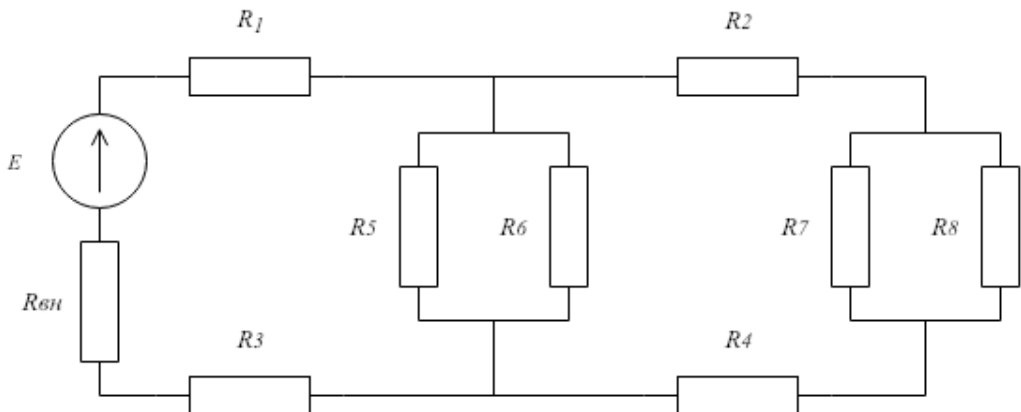
1. Продемонстрируйте, как изменить цвет графического окна пользователя.

2. Продемонстрируйте, как можно запретить/разрешить изменять размер окна пользователю.
3. Создайте поле ввода розового цвета.
4. Создайте кнопку с надписью “=”.
5. Измените название графического окна пользователя на имя “Лабораторная работа №5”.
6. Загрузите в папку “Лабораторная работа №5” любую картинку и вставьте ее вместо электрической схемы.
7. Измените название “Источник” на название “Исходные данные”.
8. Создайте окно вывода значения расчета эквивалентного сопротивления схемы.
9. Продемонстрируйте, как изменить цвет кнопки.
10. Обеспечьте передачу значения тока в окно вывода в мА.
11. Обеспечьте вывод значения тока типом шрифта “Verdana”.
12. Измените цвет поля вывода значения тока на желтый.
13. Измените размер графического окна пользователя, увеличив его ширину и высоту.
14. Добавьте подрисовочную надпись в графическом окне пользователя под электрической схемой.
15. Продемонстрируйте, как можно добавить стандартное меню окна.
16. Измените цвет текста “Сопротивления” с черного на красный.
17. Обеспечьте расположение графического окна пользователя в верхнем левом углу.
18. Обеспечьте расположение графического окна пользователя в верхнем правом углу.

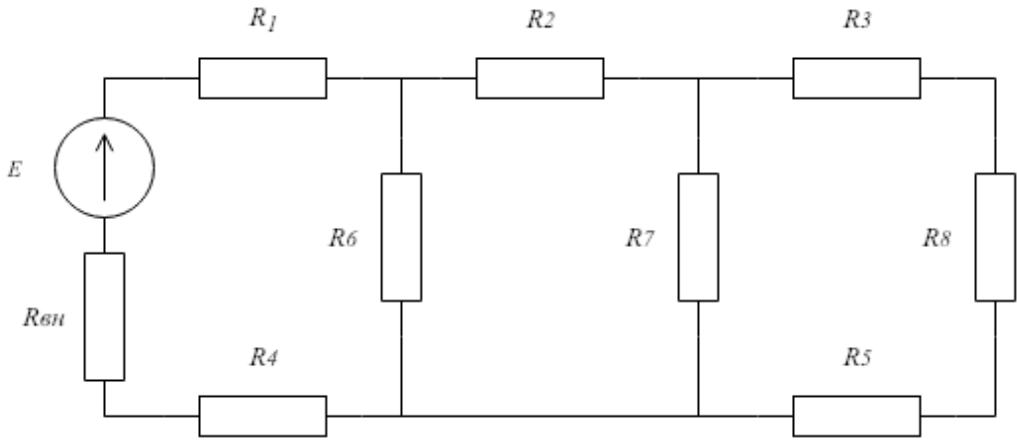
# Приложение 1

Таблица П.1 - Варианты заданий

№	Задание
1	Написать программу, строящую график функции: $f(x) = A \cdot e^{-\delta \cdot x} \cdot \sin(\omega \cdot x + \varphi)$ . Параметры $A, \delta, \omega, \varphi$ , а также диапазон построения графика $x_{min}, x_{max}$ вводятся пользователем
2	Написать программу, решающую нелинейное уравнение: $e^{-\alpha \cdot (x-\beta)^2} - \gamma x = 0$ . Параметры $\alpha, \beta, \gamma$ вводятся пользователем. Программа должна строить график функции в левой части уравнения и отмечать на нем найденный корень.
3	Вводятся координаты четырех известных точек функции: $(x_1, y_1), (x_2, y_2), (x_3, y_3), (x_4, y_4)$ . Программа должна строить график интерполяционного полинома Лагранжа.
4	Вводятся элементы матрицы размером 3х3. Написать программу, рассчитывающую определитель матрицы.
5	<p>Написать программу, решающую систему уравнений:</p> $a \cdot x + b \cdot y = c$ $d \cdot x + e \cdot y = f$ <p>Коэффициенты <math>a, b, c, d, e, f</math> вводятся пользователем.</p>
6	<p>Переходный процесс в схеме описывается дифференциальными уравнениями:</p> $E - u_C - L \frac{di_L}{dt} - i_L R_2 = 0$ $C \frac{du_C}{dt} + \frac{u_C}{R_1} - i_L = 0$ <p>Написать программу, решающую систему дифференциальных уравнений и выводящую графики <math>u_C, i_L</math>. Начальные условия - нулевые. Параметры <math>E, L, C, R_2</math> а также время расчета <math>t_{max}</math> вводятся пользователем.</p>

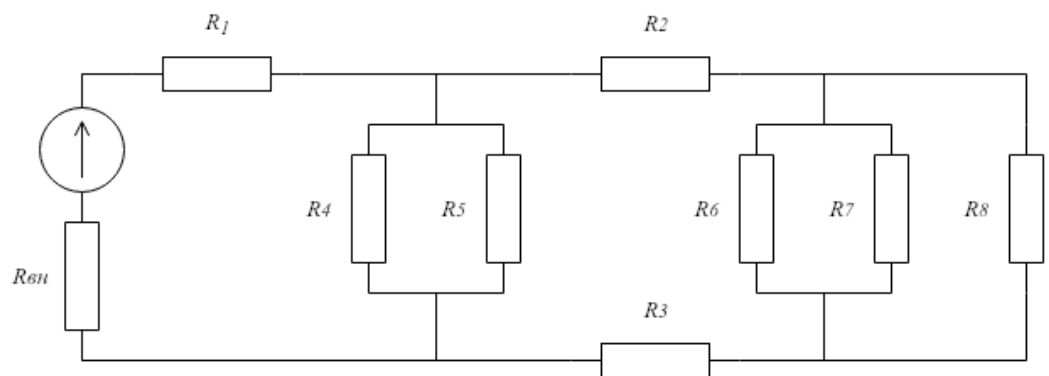
7	Вводятся элементы матрицы размером 3x3. Написать программу, рассчитывающую собственные числа этой матрицы.
8	Написать программу, решающую дифференциальное уравнение: $y' + a \cdot y = b \cdot \sin(c \cdot t)$ . Параметры $a, b, c$ , а также $y(0)$ и диапазон решения $t_{max}$ вводятся пользователем. Программа должна строить график решения.
9	Написать программу, рассчитывающую определённый интеграл функции: $f(x) = A \cdot \sin^2(\omega \cdot x)$ . Параметры $A, \omega$ , а также - диапазон расчета определённого интеграла: $x_{min}, x_{max}$ вводятся пользователем. Программа должна строить график функции и откладывать на нём значение интеграла в виде пунктирной прямой, параллельной оси абсцисс.
10	<p>Написать программу, рассчитывающую ток в схеме:</p>  <p>Значения источника напряжения и сопротивлений вводятся пользователем.</p>
11	Написать программу, строящую график функции: $f(x) = A_1 \cdot e^{-t/T1} + A_2 \cdot e^{-t/T2}$ . Параметры $A_1, A_2, T1, T2$ , а также диапазон построения графика $x_{min}, x_{max}$ вводятся пользователем
12	Написать программу, решающую нелинейное уравнение: $\ln(ax + b) - c \cdot e^{-x} = 0$ . Параметры $a, b, c$ вводятся пользователем. Программа должна строить график функции в левой части уравнения и отмечать на нем найденный корень.

13	Вводятся координаты четырех известных точек функции: $(x_1, y_1)$ , $(x_2, y_2)$ , $(x_3, y_3)$ , $(x_4, y_4)$ , а также - значение $x$ , в котором необходимо определить неизвестное значение $y$ . Программа должна определять $y$ при помощи линейной интерполяции.
14	Вводятся элементы матрицы размером $2 \times 2$ . Написать программу, рассчитывающую и выводящую обратную матрицу.
15	<p>Написать программу, решающую систему уравнений методом Крамера:</p> $a_{11} \cdot x + a_{12} \cdot y = b_1$ $a_{21} \cdot x + a_{22} \cdot y = b_2$ <p>Коэффициенты <math>a_{11}, a_{12}, a_{21}, a_{22}, b_1, b_2</math> вводятся пользователем. Программа должна выводить все найденные в процессе решения определители.</p>
16	<p>Переходный процесс в схеме описывается дифференциальными уравнениями:</p> $E - u_C - i_L R_2 - L \frac{di_L}{dt} = 0$ $C \frac{du_C}{dt} = \frac{E - u_C}{R_1} + i_L$ <p>Написать программу, решающую систему дифференциальных уравнений и выводящую графики <math>u_C</math>, <math>i_L</math>. Начальные условия - нулевые. Параметры <math>E, L, C, R_1, R_2</math> а также время расчета <math>t_{max}</math> вводятся пользователем.</p>
17	Вводятся элементы матрицы размером $2 \times 2$ . Написать программу, рассчитывающую собственные векторы матрицы.
18	Написать программу, решающую дифференциальное уравнение: $y' + a \cdot y = b \cdot e^{-ct}$ . Параметры $a, b, c$ , а также $y(0)$ и диапазон решения $t_{max}$ вводятся пользователем. Программа должна строить график решения.
19	Написать программу, рассчитывающую определённый интеграл

	<p>функции: <math>f(x) = A \cdot e^{-\alpha(x-m)^2}</math>. Параметры <math>A, \alpha, m</math>, а также - диапазон расчета определённого интеграла: <math>x_{min}, x_{max}</math> вводятся пользователем. Программа должна строить график функции и откладывать на нём значение интеграла в виде пунктирной прямой, параллельной оси абсцисс.</p>
20	<p>Написать программу, рассчитывающую ток в схеме:</p>  <p>Значения источника напряжения и сопротивлений вводятся пользователем.</p>
21	<p>Написать программу, строящую график функции: <math>f(x) = A_1 \cdot \sin(\omega_1 \cdot t) + A_2 \cdot \sin(\omega_2 \cdot t)</math>. Параметры <math>A_1, A_2, \omega_1, \omega_2</math>, а также диапазон построения графика <math>x_{min}, x_{max}</math> вводятся пользователем</p>
22	<p>Написать программу, решающую нелинейное уравнение: <math>a \cdot \arctg(b \cdot x) + c \cdot x - d = 0</math>. Параметры <math>a, b, c, d</math> вводятся пользователем. Программа должна строить график функции в левой части уравнения и отмечать на нем найденный корень.</p>
23	<p>Вводятся координаты пяти известных точек функции: <math>(x_1, y_1), (x_2, y_2), (x_3, y_3), (x_4, y_4), (x_5, y_5)</math>. Программа должна строить график интерполяционного полинома из кубических сплайнов.</p>
24	<p>Вводятся элементы матрицы размером 3x3. Написать программу, выполняющую для этой матрицы прямой ход метода Гаусса.</p>

25	<p>Написать программу, решающую систему уравнений:</p> $a_{11} \cdot x + a_{12} \cdot y = b_1$ $a_{21} \cdot x + a_{22} \cdot y = b_2$ <p>Коэффициенты <math>a_{11}, a_{12}, a_{21}, a_{22}, b_1, b_2</math> вводятся пользователем. Программа должна выводить обратную матрицу коэффициентов.</p>
26	<p>Переходный процесс в схеме описывается дифференциальными уравнениями:</p> $E - R_1 i_L - L \frac{di_L}{dt} - u_C = 0$ $i_L = C \frac{du_C}{dt} + \frac{u_C}{R_2}$ <p>Написать программу, решающую систему дифференциальных уравнений и выводящую графики <math>u_C, i_L</math>. Начальные условия - нулевые. Параметры <math>E, L, C, R_1, R_2</math> а также время расчета <math>t_{max}</math> вводятся пользователем.</p>
27	<p>Вводятся элементы матрицы размером 3x3. Написать программу, рассчитывающую ранг матрицы.</p>
28	<p>Написать программу, решающую дифференциальное уравнение: <math>y' + a \cdot y = b \cdot t + c</math>. Параметры <math>a, b, c</math>, а также <math>y(0)</math> и диапазон решения <math>t_{max}</math> вводятся пользователем. Программа должна строить график решения.</p>
29	<p>Написать программу, рассчитывающую определённый интеграл функции: <math>f(x) = e^{-b \cdot x} \cdot \cos^2(c \cdot x)</math>. Параметры <math>b, c</math>, а также - диапазон расчета определённого интеграла: <math>x_{min}, x_{max}</math> вводятся пользователем. Программа должна строить график функции и откладывать на нём значение интеграла в виде пунктирной прямой, параллельной оси абсцисс.</p>
30	<p>Написать программу, рассчитывающую ток в схеме:</p>





Значения источника напряжения и сопротивлений вводятся пользователем.