Министерство науки и высшего образования Российской Федерации
Федеральное государственное автономное образовательное учреждение высшего образования
«СЕВЕРО-КАВКАЗСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»

Институт перспективной инженерии
Департамент цифровых, робототехнических систем и электроники

ОТЧЕТ
ПО ЛАБОРАТОРНОЙ РАБОТЕ №4.8
дисциплины «Объектно-ориентированное программирование»
Вариант 9

Выполнил:
Иващенко Олег Андреевич
3 курс, группа ИВТ-б-о-22-1,
09.03.02 «Информационные и вычислительные машины»,
направленность (профиль)
«Программное обеспечение средств вычислительной техники и автоматизированных систем»

_____
(подпись)

Руководитель практики:
Воронкин Роман Александрович, доцент департамента цифровых, робототехнических систем и электроники

_____
(подпись)

Отчет защищен с оценкой _____ Дата защиты_____

Ставрополь, 2024 г.

Тема: «Обработка событий и рисование в Tkinter»

Цель: Приобретение навыков улучшения графического интерфейса пользователя GUI с помощью обработки событий и рисования, реализованных в пакете Tkinter языка программирования Python версии 3.x.

Порядок выполнения работы:

Задача 1. Напишите программу, состоящую из двух списков Listbox. В первом будет, например, перечень товаров, заданный программно. Второй изначально пуст, пусть это будет перечень покупок. При клике на одну кнопку товар должен переходить из одного списка в другой. При клике на вторую кнопку – возвращаться (человек передумал покупать). Предусмотрите возможность множественного выбора элементов списка и их перемещения.

Таблица 1 – Листинг программы

```
#!/usr/bin/env python3
# -*- coding: utf-8 -*-

"""
Напишите программу, состоящую из двух списков Listbox. В первом будет,
например, перечень товаров, заданный программно. Второй изначально пуст,
пусть это будет перечень покупок. При клике на одну кнопку товар должен
переходить из одного списка в другой. При клике на вторую
кнопку – возвращаться (человек передумал покупать). Предусмотрите возможность
множественного выбора элементов списка и их перемещения.
"""

from tkinter import *

def move_to_cart():
    selected_items = listbox1.curselection()

    for index in reversed(selected_items):
        item = listbox1.get(index)
        listbox2.insert(END, item)
        listbox1.delete(index)

def move_to_products():
    selected_items = listbox2.curselection()

    for index in reversed(selected_items):
        item = listbox2.get(index)
        listbox1.insert(END, item)
```

```
        listbox2.delete(index)

root = Tk()
root.title("Список покупок")
root.geometry("400x200")

main_frame = Frame(root)
main_frame.pack(pady=10)

listbox1 = Listbox(main_frame, selectmode=MULTIPLE)
listbox1.pack(side=LEFT, padx=10)

button_frame = Frame(main_frame)
button_frame.pack(side=LEFT, padx=10)

add_button = Button(button_frame, text=">>>", command=move_to_cart)
add_button.pack(pady=5)

remove_button = Button(button_frame, text="<<<", command=move_to_products)
remove_button.pack(pady=5)

listbox2 = Listbox(main_frame, selectmode=MULTIPLE)
listbox2.pack(side=RIGHT, padx=10)

products = ["Яблоки", "Бананы", "Молоко", "Хлеб", "Сыр", "Мясо", "Рыба"]
for product in products:
    listbox1.insert(END, product)

root.mainloop()
```

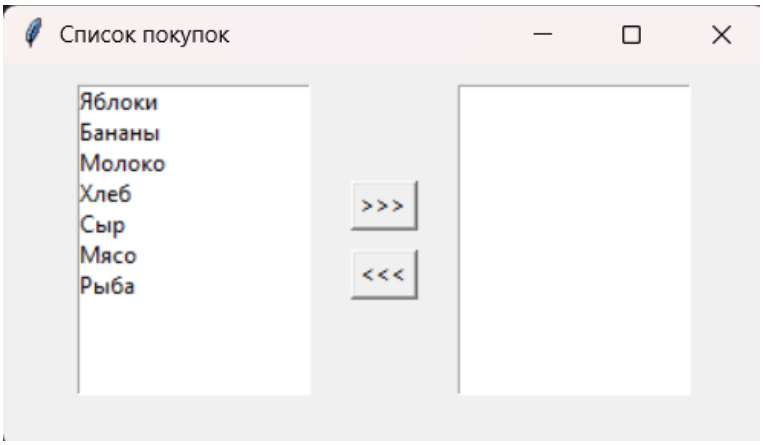


Рисунок 1.1 – Окно программы

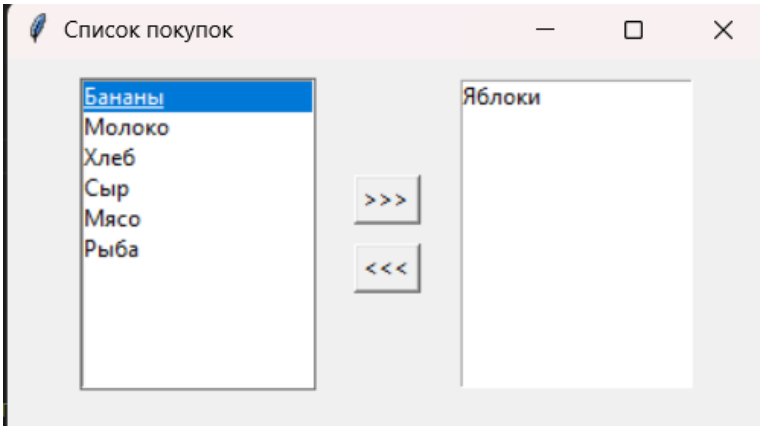


Рисунок 1.2 – Перенос объектов во второй список

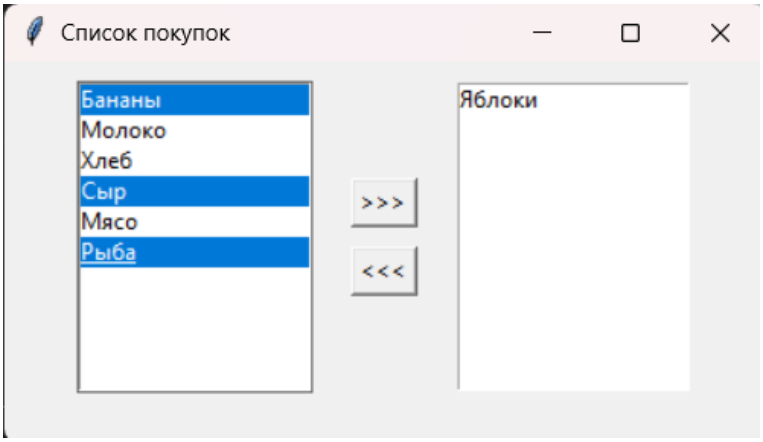


Рисунок 1.3 – Множественный выбор

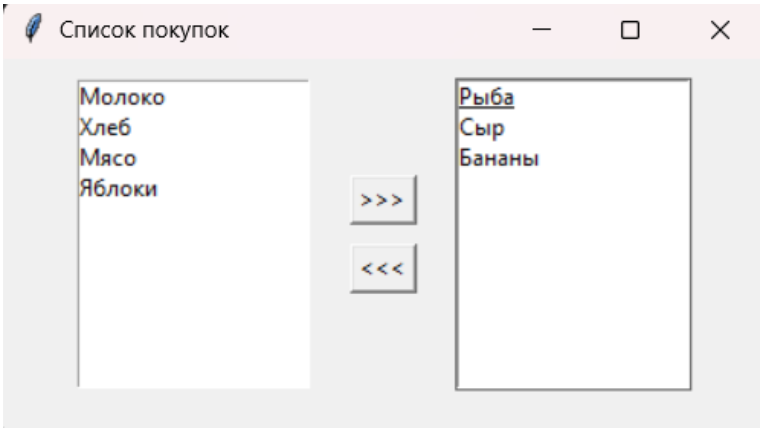


Рисунок 1.4 – Обратное перемещение в первый список

Задача 2. Напишите программу по следующему описанию. Нажатие Enter в однострочном текстовом поле приводит к перемещению текста из

него в список (экземпляр Listbox). При двойном клике (<Double-Button-1>) по элементу-строке списка, она должна копироваться в текстовое поле.

Таблица 2 – Листинг программы

```
#!/usr/bin/env python3
# -*- coding: utf-8 -*-

"""
Решите задачу: напишите программу по следующему описанию.
Нажатие Enter в однострочном текстовом поле приводит к перемещению текста
из него в список (экземпляр Listbox). При двойном клике (<Double-Button-1>)
по элементу-строке списка, она должна копироваться в текстовое поле.
"""

from tkinter import *

def add_to_list(event=None):
    text = entry.get()
    if text:
        listbox.insert(END, text)
        entry.delete(0, END)

def copy_to_entry(event):
    selected_index = listbox.curselection()
    if selected_index:
        text = listbox.get(selected_index)
        entry.delete(0, END)
        entry.insert(0, text)

root = Tk()
root.title("Текст и список")
root.geometry("400x300")

entry = Entry(root, width=40)
entry.pack(pady=10)

entry.bind("<Return>", add_to_list)

listbox = Listbox(root, width=40)
listbox.pack(pady=10)

listbox.bind("<Double-Button-1>", copy_to_entry)

root.mainloop()
```

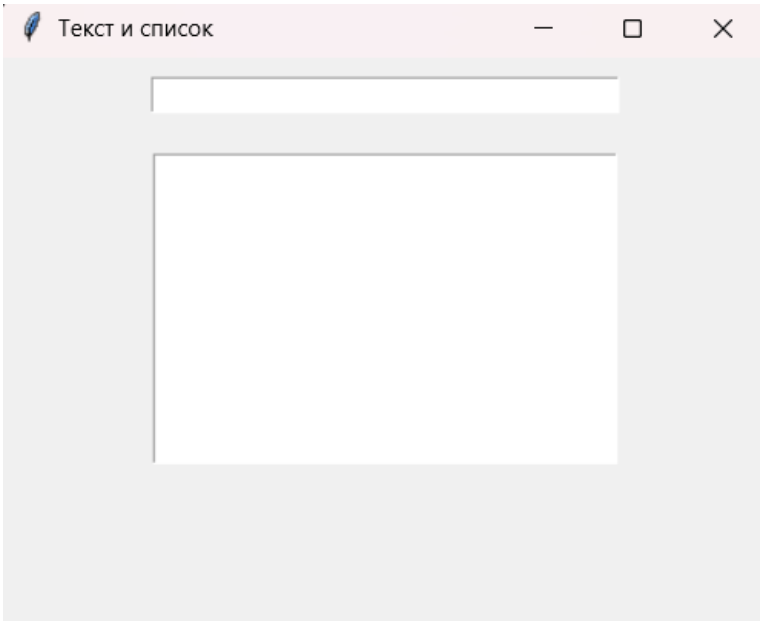


Рисунок 2.1 – Окно программы

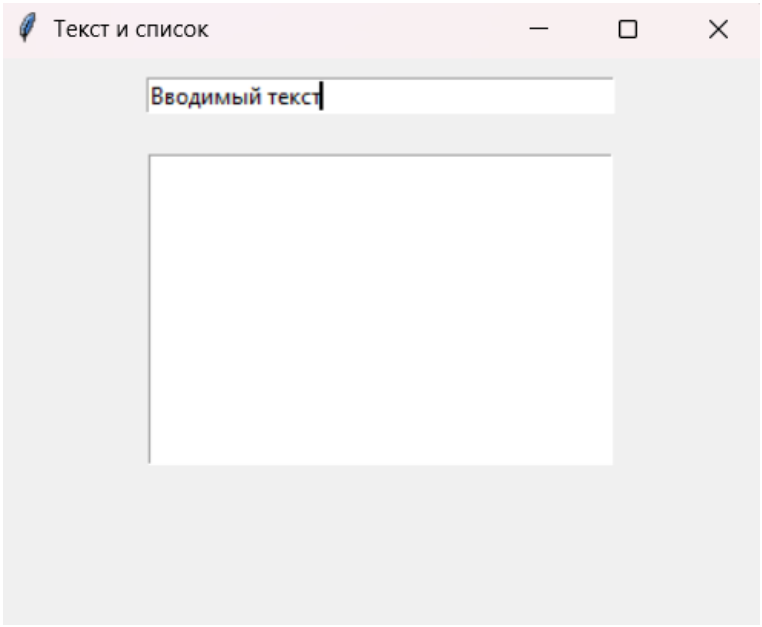


Рисунок 2.2 – Ввод текста в текстовое поле

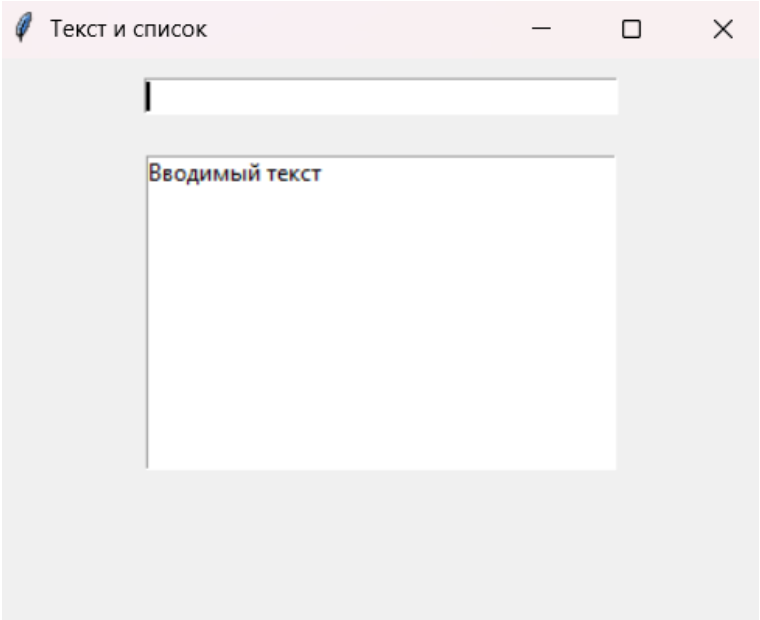


Рисунок 2.3 – Перенос текста в список

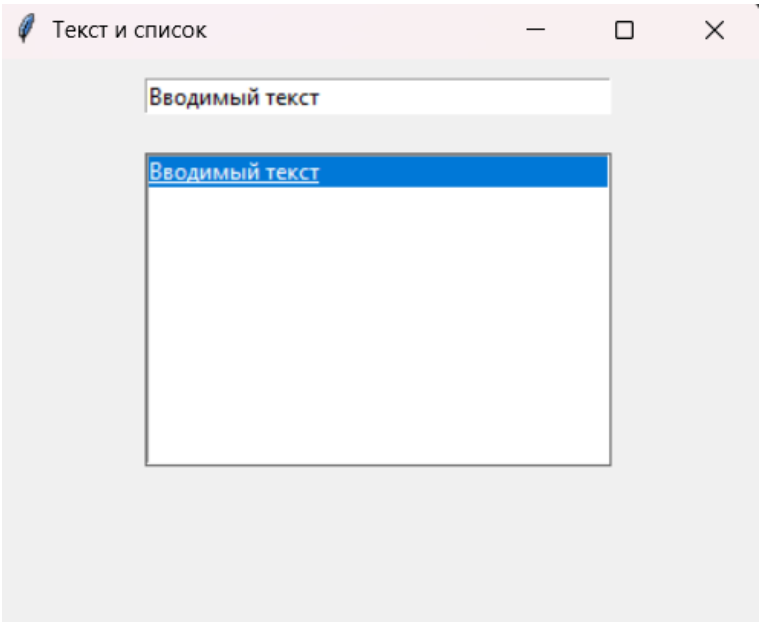


Рисунок 2.4 – Копирование выбранного текста в текстовое поле

Задача 3. Напишите программу по описанию. Размеры многострочного текстового поля определяются значениями, введенными в однострочные текстовые поля. Изменение размера происходит при нажатии мышью на кнопку, а также при нажатии клавиши Enter. Цвет фона экземпляра Text светлосерый (lightgrey), когда поле не в фокусе, и белый, когда имеет фокус. Событие получения фокуса обозначается как <FocusIn>, потери – как

<FocusOut>. Для справки: фокус перемещается по виджетам при нажатии Tab, Ctrl+Tab, Shift+Tab, а также при клике по ним мышью (к кнопкам последнее не относится).

Таблица 3 – Листинг программы

```
#!/usr/bin/env python3
# -*- coding: utf-8 -*-

"""
Напишите программу по описанию. Размеры многострочного текстового поля определяются значениями, введенными в однострочные текстовые поля. Изменение размера происходит при нажатии мышью на кнопку, а также при нажатии клавиши Enter. Цвет фона экземпляра Text светлосерый (lightgrey), когда поле не в фокусе, и белый, когда имеет фокус. Событие получения фокуса обозначается как <FocusIn>, потери – как <FocusOut>. Для справки: фокус перемещается по виджетам при нажатии Tab, Ctrl+Tab, Shift+Tab, а также при клике по ним мышью (к кнопкам последнее не относится).
"""

from tkinter import *
from tkinter import messagebox

def resize_text(event=None):
    try:
        width = int(width_entry.get())
        height = int(height_entry.get())

        text_field.config(width=width, height=height)

        text_frame.config(width=width * 10, height=height * 20)
    except ValueError:
        messagebox.showerror("Ошибка", "Введите корректные числа!")

def on_focus_in(event):
    text_field.config(bg="white")

def on_focus_out(event):
    text_field.config(bg="lightgrey")

root = Tk()
root.title("Изменение размеров текстового поля")
root.geometry("400x300")

input_frame = Frame(root)
input_frame.pack(pady=10)

width_label = Label(input_frame, text="Ширина:")
width_label.pack(side=LEFT, padx=5)
width_entry = Entry(input_frame, width=10)
width_entry.pack(side=LEFT, padx=5)
```

```
height_label = Label(input_frame, text="Высота:")
height_label.pack(side=LEFT, padx=5)
height_entry = Entry(input_frame, width=10)
height_entry.pack(side=LEFT, padx=5)

resize_button = Button(root, text="Изменить размер", command=resize_text)
resize_button.pack(pady=5)

text_frame = Frame(root)
text_frame.pack(pady=10)

text_frame.pack_propagate(False)

text_field = Text(text_frame, bg="lightgrey")
text_field.pack(fill=BOTH, expand=True)

text_field.bind("<FocusIn>", on_focus_in)
text_field.bind("<FocusOut>", on_focus_out)

width_entry.bind("<Return>", resize_text)
height_entry.bind("<Return>", resize_text)

root.mainloop()
```

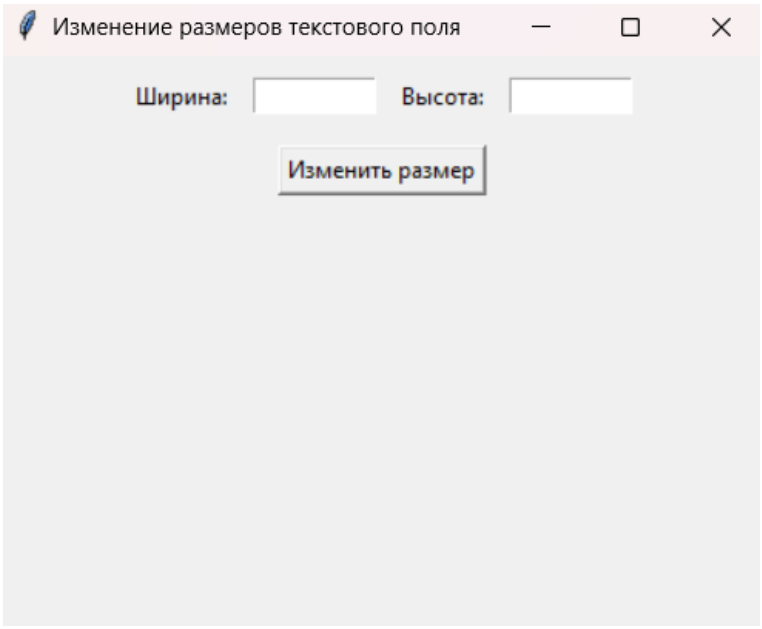


Рисунок 3.1 – Окно программы

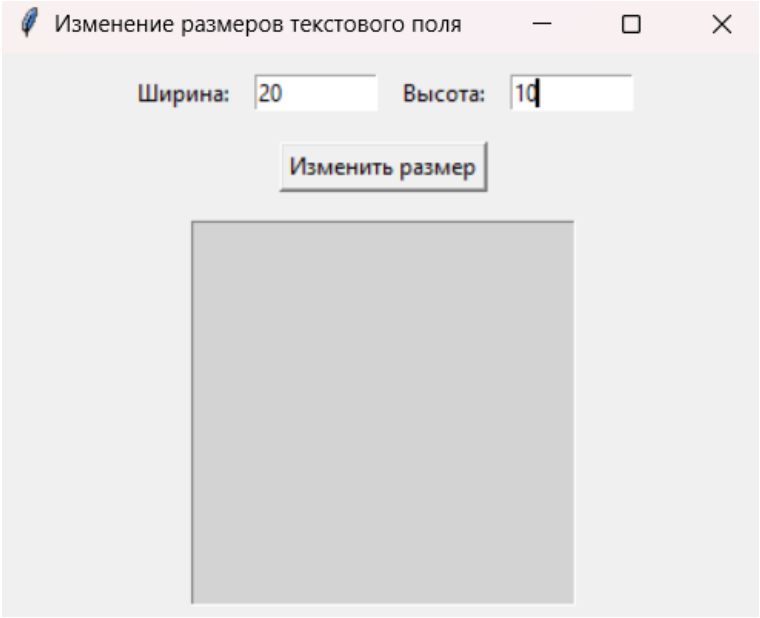


Рисунок 3.2 – Создание текстового поля указанного размера

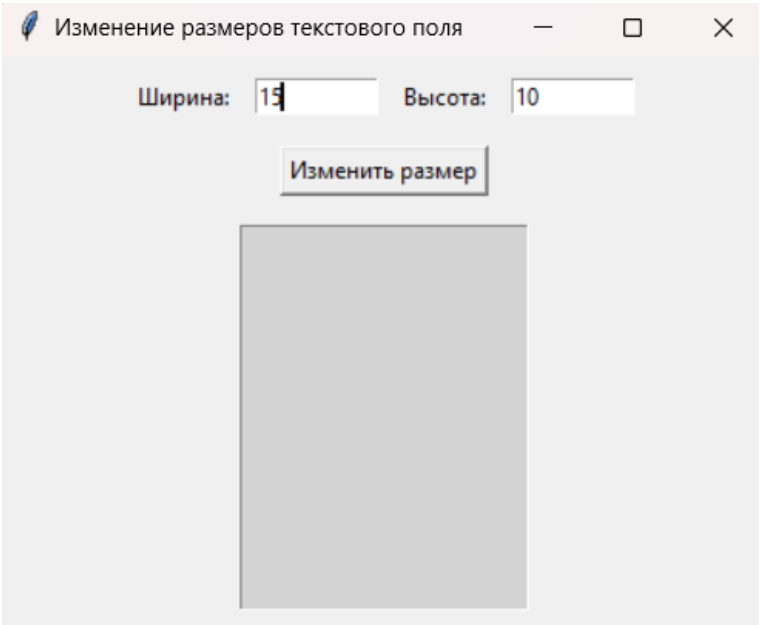


Рисунок 3.3 – Изменение размера текстового поля (при нажатии на клавишу Enter)

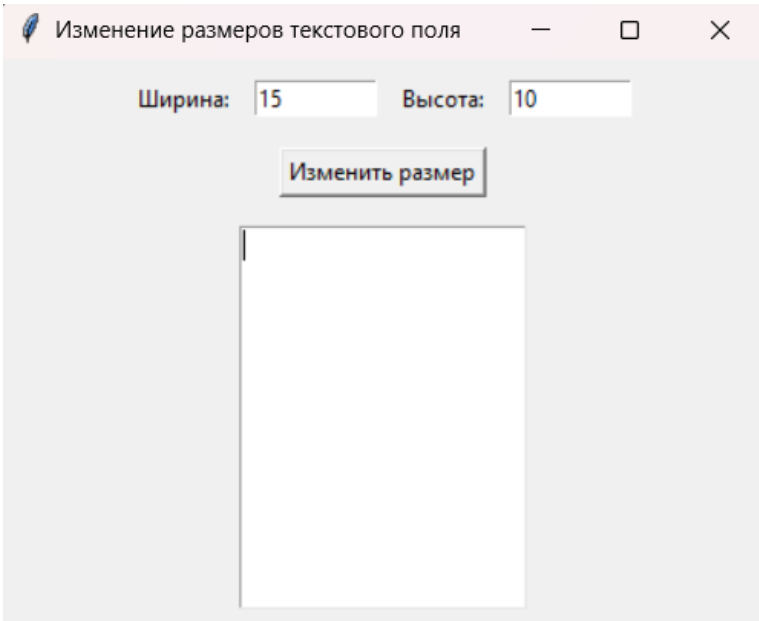
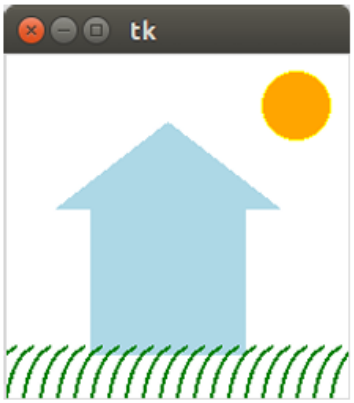


Рисунок 3.4 – Смена фона текстового поля на белый цвет при фокусе на нём

Задача 4. Создайте на холсте подобное изображение:



Для создания травы используйте цикл.

Таблица 4 – Листинг программы

```
#!/usr/bin/env python3
# -*- coding: utf-8 -*-

"""
Создайте на холсте изображение домика, солнца и травы. Для создания
травы необходимо использовать цикл.
"""

from tkinter import *

root = Tk()
```

```
root.title("Картинка")
root.geometry("300x350")

canvas = Canvas(root, bg="white", width=500, height=400)
canvas.pack()

canvas.create_rectangle(100, 200, 200, 300, fill="lightblue", outline="lightblue")
canvas.create_polygon(75, 200, 225, 200, 150, 150, fill="lightblue", outline="lightblue")
canvas.create_oval(200, 50, 250, 100, fill="orange", outline="orange")

for i in range(0, 500, 10):
    canvas.create_line(i, 350, i + 10, 320, fill="green", smooth=True)

root.mainloop()
```

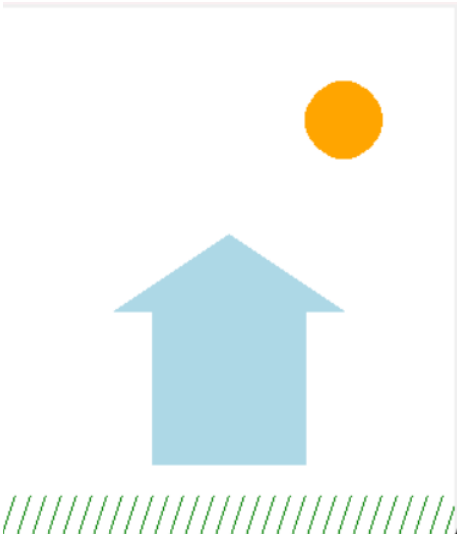


Рисунок 4 – Вывод программы

Задача 5. В данной программе создается анимация круга, который движется от левой границы холста до правой. Выражение c.coords(ball) возвращает список текущих координат объекта (в данном случае это ball). Третий элемент списка соответствует его второй координате x. Метод after вызывает функцию, переданную вторым аргументом, через количество миллисекунд, указанных первым аргументом. Изучите приведенную программу и самостоятельно запрограммируйте постепенное движение фигуры в ту точку холста, где пользователь кликает левой кнопкой мыши. Координаты события хранятся в его атрибутах x и y (event.x, event.y).

Таблица 5 – Листинг программы

```
#!/usr/bin/env python3
# -*- coding: utf-8 -*-

"""
В данной программе создается анимация круга, который движется от левой границы
холста до правой. Выражение c.coords(ball) возвращает список текущих координат
объекта (в данном случае это ball). Третий элемент списка соответствует его
второй координате x. Метод after вызывает функцию, переданную вторым
аргументом, через количество миллисекунд, указанных первым аргументом. Изучите
приведенную программу и самостоятельно запрограммируйте постепенное движение
фигуры в ту точку холста, где пользователь кликает левой кнопкой мыши.
Координаты события хранятся в его атрибутах x и y (event.x, event.y).
"""

from tkinter import *

def on_click(event):
    global target_x, target_y
    target_x, target_y = event.x, event.y
    move_towards_target()

def move_towards_target():
    global target_x, target_y

    current_x1, current_y1, current_x2, current_y2 = c.coords(ball)
    current_x = (current_x1 + current_x2) / 2
    current_y = (current_y1 + current_y2) / 2

    dx = target_x - current_x
    dy = target_y - current_y

    if abs(dx) > 1 or abs(dy) > 1:
        c.move(ball, dx * 0.1, dy * 0.1)
        root.after(10, move_towards_target)

root = Tk()
root.title("Движение к клику")

c = Canvas(root, width=300, height=200, bg="white")
c.pack()

ball = c.create_oval(0, 100, 40, 140, fill='green')

target_x, target_y = 0, 0

c.bind("<Button-1>", on_click)

root.mainloop()
```

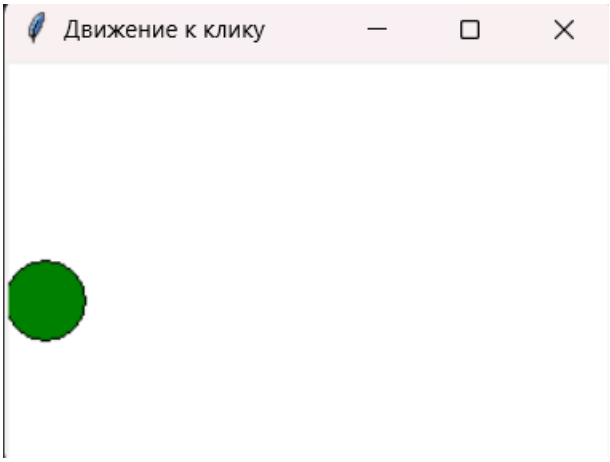


Рисунок 5.1 – Окно программы при запуске

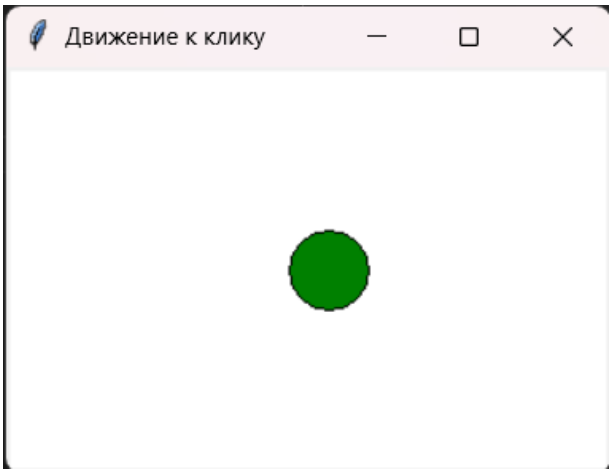


Рисунок 5.2 – Движение круга к точке клика мышью

Ответы на контрольные вопросы:

1. Каково назначение виджета ListBox?

ListBox (список) — это виджет, который позволяет отображать список элементов, из которых пользователь может выбирать один или несколько. Он используется для:

- Отображения списка данных (например, файлов, товаров, имен).
- Выбора одного или нескольких элементов (в зависимости от настроек).
- Взаимодействия с пользователем через выбор элементов.

2. Каким образом осуществляется связывание событие или действие с виджетом Tkinter?

Связывание события или действия с виджетом осуществляется с помощью метода bind. Этот метод принимает два аргумента:

- Строку с описанием события (например, <Button-1> для клика левой кнопкой мыши).
- Функцию-обработчик, которая будет вызвана при возникновении события.

3. Какие существуют типы событий в Tkinter? Приведите примеры.

В Tkinter существует множество типов событий. Вот основные категории и примеры:

События мыши:

- <Button-1> — клик левой кнопкой мыши.
- <Button-3> — клик правой кнопкой мыши.
- <Motion> — движение мыши.
- <Double-Button-1> — двойной клик левой кнопкой мыши

События клавиатуры:

- <KeyPress> — нажатие любой клавиши.
- <Return> — нажатие клавиши Enter.
- <Escape> — нажатие клавиши Esc.

События фокуса:

- <FocusIn> — виджет получил фокус.
- <FocusOut> — виджет потерял фокус.

События окна:

- <Configure> — изменение размеров окна.
- <Destroy> — закрытие окна.

4. Как обрабатываются события в Tkinter?

События в Tkinter обрабатываются с помощью функций-обработчиков, которые связываются с виджетами через метод bind. Когда происходит событие (например, клик мыши или нажатие клавиши), Tkinter вызывает соответствующую функцию-обработчик.

5. Как обрабатываются события мыши в Tkinter?

События мыши обрабатываются с помощью функций-обработчиков, которые связываются с виджетами через метод bind. Каждое событие мыши передает объект event, содержащий координаты клика (event.x, event.y) и другую информацию.

6. Каким образом можно отображать графические примитивы в Tkinter?

Графические примитивы (например, линии, прямоугольники, круги) отображаются на холсте (Canvas) с помощью методов:

- create_line — линия.
- create_rectangle — прямоугольник.
- create_oval — круг или эллипс.
- create_polygon — многоугольник.
- create_text — текст.

7. Перечислите основные методы для отображения графических примитивов в Tkinter.

Основные методы для отображения графических примитивов на холсте (Canvas):

- create_line(x1, y1, x2, y2, options) — линия.
- create_rectangle(x1, y1, x2, y2, options) — прямоугольник.
- create_oval(x1, y1, x2, y2, options) — круг или эллипс.
- create_polygon(x1, y1, x2, y2, ..., options) — многоугольник.
- create_text(x, y, text="Текст", options) — текст.

- create_arc(x1, y1, x2, y2, options) — дуга.

8. Каким образом можно обратиться к ранее созданным фигурам на холсте?

К ранее созданным фигурам на холсте можно обратиться с помощью их идентификаторов (ID), которые возвращаются методами создания примитивов (например, create_rectangle, create_oval). Эти ID можно использовать для изменения или удаления фигур.

9. Каково назначение тэгов в Tkinter?

Тэги в Tkinter используются для:

- Группировки нескольких виджетов или графических объектов (например, для одновременного изменения их свойств).
- Применения стилей или обработки событий к группе объектов.
- Упрощения управления объектами (например, удаление или изменение всех объектов с определенным тэгом).

Вывод: В процессе выполнения лабораторной работы были приобретены навыки улучшения графического интерфейса пользователя GUI с помощью обработки событий и рисования, реализованных в пакете Tkinter языка программирования Python версии 3.x.

Ссылка на репозиторий GitHub:

IUnnamedUserI/OOP_8: Объектно-ориентированное программирование. Лабораторная работа №8