Національний технічний університет України «Київський політехнічний інститут імені Ігоря Сікорського»

Факультет інформатики та обчислювальної техніки

Кафедра обчислювальної техніки

Програмування

Лабораторна робота №8

«Розробка програми з графічним інтерфейсом на основі бібліотеки tkinter»

Виконав:
студент групи ІО-41
Давидчук А. М.
Залікова книжка № 4108

Перевірив
Пономаренко А. М.

Київ-2024

Тема: «Розробка програми з графічним інтерфейсом на основі бібліотеки tkinter».

Мета: ознайомитися з організацією графічного інтерфейсу на основі бібліотеки tkinter.

Короткі теоретичні відомості:

Tk() – створює головне вікно (кореневий елемент інтерфейсу) програми на основі tkinter.
Toplevel – віджет для створення додаткових вікон поверх основного. Дозволяє мати кілька вікон, окрім головного.
LabelFrame – контейнер, що об'єднує групу віджетів, оточуючи їх рамкою з опціональною назвою.
Label – простий текстовий віджет для виведення статичного тексту.
Entry – віджет для введення тексту користувачем.
Button – натискна кнопка. При натисканні виконує зв'язану з нею функцію (команду).
Menu, Menu.add_command – Меню на панелі інструментів вікна. add_command додає нові пункти меню, які при виборі запускають певну дію.
messagebox – модуль tkinter для виведення стандартних діалогових вікон (наприклад, повідомлення про помилку, попередження, інформацію).
filedialog – модуль tkinter для відкриття діалогових вікон вибору файлів (збереження або відкриття).
set – стандартний тип даних у Python для збереження унікальних елементів та виконання операцій над множинами (перетин, об'єднання, різниця).
методи множин (union, intersection, difference) – стандартні методи для операцій над множинами:
- union() – об'єднання множин
- intersection() – перетин множин
- difference() – різниця множин

range(min_val, max_val+1) – вбудована функція Python, що генерує послідовність цілих чисел у заданому діапазоні.
random.sample(послідовність, k) – функція з модуля random для вибірки k унікальних елементів випадковим чином із зазначеної послідовності.
winfo_exists() – метод віджетів tkinter для перевірки, чи існує ще конкретне вікно (чи воно не було закрите).
command=функція – параметр віджетів (наприклад, Button), що задає функцію зворотного виклику (callback), яка буде виконана при натисканні кнопки чи іншій взаємодії.

Завдання:

8	$D = \left(\left(A \cap \overline{B}\right) \cup \left(B \cap \overline{A}\right)\right) \cap \left(C \cup B\right) \cap C$ $X = \overline{F} \cup \overline{D}$

Код:

```
import random, tkinter as tk
from tkinter import Toplevel, Menu, filedialog, messagebox

class Window1:
```

```
    def __init__(self, master):

        self.master = master
        self.master.title("Вікно №1")

        self.menu_bar = Menu(self.master)
        self.window_menu = Menu(self.menu_bar, tearoff=0)
        self.window_menu.add_command(label="Вікно №2",
command=self.open_window2)
        self.window_menu.add_command(label="Вікно №3",
command=self.open_window3)
        self.menu_bar.add_cascade(label="Вікна", menu=self.window_menu)

        self.master.config(menu=self.menu_bar)

        self.frame_student_info = tk.LabelFrame(self.master, text="Дані
студента", padx=5, pady=5)
        self.frame_student_info.pack(padx=10, pady=10, fill='x')

        tk.Label(self.frame_student_info, text="П.І.Б.:").grid(row=0,
column=0, sticky='w')
        self.entry_name = tk.Entry(self.frame_student_info)
        self.entry_name.grid(row=0, column=1, sticky='w', padx=5)

        tk.Label(self.frame_student_info, text="Номер групи:").grid(row=1,
column=0, sticky='w')
        self.entry_group = tk.Entry(self.frame_student_info)
        self.entry_group.grid(row=1, column=1, sticky='w', padx=5)

        tk.Label(self.frame_student_info, text="Номер у списку:").grid(row=2,
column=0, sticky='w')
        self.entry_number = tk.Entry(self.frame_student_info)
        self.entry_number.grid(row=2, column=1, sticky='w', padx=5)

        self.frame_sets = tk.LabelFrame(self.master, text="Параметри множин",
padx=5, pady=5)
        self.frame_sets.pack(padx=10, pady=10, fill='x')

        tk.Label(self.frame_sets, text="К-сть ел-тів A:").grid(row=0,
column=0)
        self.entry_size_A = tk.Entry(self.frame_sets, width=5)
        self.entry_size_A.grid(row=0, column=1, padx=5)

        tk.Label(self.frame_sets, text="К-сть ел-тів B:").grid(row=1,
column=0)
        self.entry_size_B = tk.Entry(self.frame_sets, width=5)
        self.entry_size_B.grid(row=1, column=1, padx=5)

        tk.Label(self.frame_sets, text="К-сть ел-тів C:").grid(row=2,
column=0)
        self.entry_size_C = tk.Entry(self.frame_sets, width=5)
        self.entry_size_C.grid(row=2, column=1, padx=5)

        tk.Label(self.frame_sets, text="Мін. значення:").grid(row=3,
column=0)
```

```
        self.entry_min_val = tk.Entry(self.frame_sets, width=5)
        self.entry_min_val.grid(row=3, column=1, padx=5)

        tk.Label(self.frame_sets, text="Макс. значення:").grid(row=4,
column=0)
        self.entry_max_val = tk.Entry(self.frame_sets, width=5)
        self.entry_max_val.grid(row=4, column=1, padx=5)

        self.button_generate_sets = tk.Button(self.frame_sets,
text="Згенерувати множини", command=self.generate_sets)
        self.button_generate_sets.grid(row=5, column=0, columnspan=2, pady=5)

        self.frame_manual = tk.LabelFrame(self.master, text="Ручний ввід
множин", padx=5, pady=5)
        self.frame_manual.pack(padx=10, pady=10, fill='x')

        tk.Label(self.frame_manual, text="A:").grid(row=0, column=0)
        self.entry_A = tk.Entry(self.frame_manual, width=30)
        self.entry_A.grid(row=0, column=1, padx=5)

        tk.Label(self.frame_manual, text="B:").grid(row=1, column=0)
        self.entry_B = tk.Entry(self.frame_manual, width=30)
        self.entry_B.grid(row=1, column=1, padx=5)

        tk.Label(self.frame_manual, text="C:").grid(row=2, column=0)
        self.entry_C = tk.Entry(self.frame_manual, width=30)
        self.entry_C.grid(row=2, column=1, padx=5)

        self.button_set_manual = tk.Button(self.frame_manual,
text="Встановити вручну", command=self.set_manual_sets)
        self.button_set_manual.grid(row=3, column=0, columnspan=2, pady=5)

        self.frame_result = tk.LabelFrame(self.master, text="Результати",
padx=5, pady=5)
        self.frame_result.pack(padx=10, pady=10, fill='x')

        self.A = set()
        self.B = set()
        self.C = set()
        self.min_val = None
        self.max_val = None

        self.window2 = None
        self.window3 = None

    def open_window2(self):
        if self.window2 is None or not self.window2.top.winfo_exists():
            if self.min_val is None or self.max_val is None:
                messagebox.showwarning("Попередження", "Спершу задайте
діапазон та множини."); return
            self.window2 = Window2(self.master, self.A, self.B, self.C,
self.min_val, self.max_val)
        else:
            self.window2.top.lift()

    def open_window3(self):
```

```
        if self.window3 is None or not (self.window3.top.winfo_exists() if
self.window3 else False):
            if self.window2 is None or not (self.window2.top.winfo_exists()
if self.window2 else False):
                messagebox.showwarning("Попередження", "Спершу обчисліть D у
вікні №2"); return
            D = self.window2.D
            if D is None:
                messagebox.showwarning("Попередження", "Спершу обчисліть D у
вікні №2"); return
            self.window3 = Window3(self.master, D, self.min_val,
self.max_val)
        else:
            self.window3.top.lift()

    def generate_sets(self):
        try:
            size_A = int(self.entry_size_A.get())
            size_B = int(self.entry_size_B.get())
            size_C = int(self.entry_size_C.get())
            min_val = int(self.entry_min_val.get())
            max_val = int(self.entry_max_val.get())
        except ValueError:
            messagebox.showerror("Помилка", "Невірні числові дані")
            return
        if min_val > max_val:
            messagebox.showerror("Помилка", "Мін. значення більше за макс.")
            return

        self.min_val = min_val
        self.max_val = max_val

        universe = list(range(min_val, max_val+1))
        if size_A > len(universe) or size_B > len(universe) or size_C >
len(universe):
            messagebox.showerror("Помилка", "Розмір множини більший за
кількість доступних елементів у діапазоні.")
            return

        self.A = set(random.sample(universe, size_A))
        self.B = set(random.sample(universe, size_B))
        self.C = set(random.sample(universe, size_C))

        self.entry_A.delete(0, tk.END)
        self.entry_B.delete(0, tk.END)
        self.entry_C.delete(0, tk.END)

        self.entry_A.insert(0, ", ".join(map(str, sorted(self.A))))
        self.entry_B.insert(0, ", ".join(map(str, sorted(self.B))))
        self.entry_C.insert(0, ", ".join(map(str, sorted(self.C))))

    def set_manual_sets(self):
        def parse_set(entry_text):
            entry_text = entry_text.strip()
            if not entry_text:
                return set()
```

```
            return set(map(int, entry_text.replace(" ", "").split(",")))
        try:
            self.A = parse_set(self.entry_A.get())
            self.B = parse_set(self.entry_B.get())
            self.C = parse_set(self.entry_C.get())
            if self.min_val is None or self.max_val is None:
                messagebox.showinfo("Увага", "Задайте спершу діапазон
мін/макс значень!")
                return
            for s in [self.A, self.B, self.C]:
                for x in s:
                    if x < self.min_val or x > self.max_val:
                        messagebox.showerror("Помилка", "Елементи множин не
входять у заданий діапазон!")
                        return
            messagebox.showinfo("Успіх", "Множини встановлені вручну")
        except ValueError:
            messagebox.showerror("Помилка", "Невірний формат введення
множин")

class Window2:

    def __init__(self, master, A, B, C, min_val, max_val):
        self.top = Toplevel(master)
        self.top.title("Вікно №2")

        self.A = A
        self.B = B
        self.C = C
        self.min_val = min_val
        self.max_val = max_val

        self.U = set(range(self.min_val, self.max_val+1))

        self.D = None

        self.frame_sets = tk.LabelFrame(self.top, text="Множини A, B, C",
padx=5, pady=5)
        self.frame_sets.pack(padx=10, pady=10, fill='x')

        tk.Label(self.frame_sets, text="A:").grid(row=0, column=0,
sticky='w')
        self.label_A = tk.Label(self.frame_sets, text=", ".join(map(str,
sorted(self.A))))
        self.label_A.grid(row=0, column=1, sticky='w')

        tk.Label(self.frame_sets, text="B:").grid(row=1, column=0,
sticky='w')
        self.label_B = tk.Label(self.frame_sets, text=", ".join(map(str,
sorted(self.B))))
        self.label_B.grid(row=1, column=1, sticky='w')

        tk.Label(self.frame_sets, text="C:").grid(row=2, column=0,
sticky='w')
        self.label_C = tk.Label(self.frame_sets, text=", ".join(map(str,
sorted(self.C))))
```

```
        self.label_C.grid(row=2, column=1, sticky='w')

        self.steps_desc = [
            "U = {min_val,...,max_val}",
            "Ā = U \\ A",
            "B̄ = U \\ B",
            "A∩B̄",
            "B∩Ā",
            "(A∩B̄) ∪ (B∩Ā)",
            "C ∪ B",
            "((A∩B̄) ∪ (B∩Ā)) ∩ (C ∪ B)",
            "D = [((A∩B̄) ∪ (B∩Ā)) ∩ (C ∪ B)] ∩ C"
        ]
        self.current_step = 0

        # Проміжні змінні
        self.A_bar = None
        self.B_bar = None
        self.step_AintBbar = None
        self.step_BintAbar = None
        self.step_sym_diff_AB = None
        self.step_CuB = None
        self.step_intermediate = None

        self.frame_step = tk.LabelFrame(self.top, text="Покрокове виконання",
padx=5, pady=5)
        self.frame_step.pack(padx=10, pady=10, fill='x')

        self.label_current_step = tk.Label(self.frame_step, text="Натисніть
'Наступний крок' для початку")
        self.label_current_step.pack()

        self.button_next_step = tk.Button(self.frame_step, text="Наступний
крок", command=self.do_next_step)
        self.button_next_step.pack(pady=5)

        self.frame_operations = tk.LabelFrame(self.top, text="Проміжні
результати", padx=5, pady=5)
        self.frame_operations.pack(padx=10, pady=10, fill='x')

        self.label_operation_result = tk.Label(self.frame_operations,
text="")
        self.label_operation_result.pack()

        self.frame_save = tk.LabelFrame(self.top, text="Результат D", padx=5,
pady=5)
        self.frame_save.pack(padx=10, pady=10, fill='x')

        self.label_D = tk.Label(self.frame_save, text="D: поки що не
обчислено")
        self.label_D.pack()

        self.button_save_D = tk.Button(self.frame_save, text="Зберегти D у
файл", command=self.save_D_to_file, state='disabled')
        self.button_save_D.pack(pady=5)
```

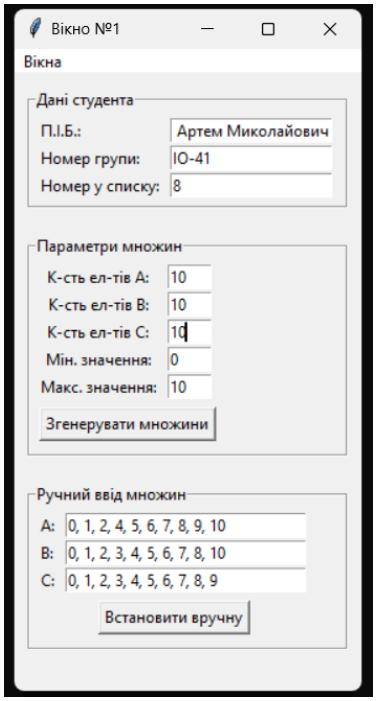
```
        self.frame_expr_img = tk.LabelFrame(self.top, text="Вираз 1", padx=5,
pady=5)
        self.frame_expr_img.pack(padx=10, pady=10, fill='x')

        tk.Label(self.frame_expr_img, text="D = (((A ∩ B̄) ∪ (B ∩ Ā)) ∩ (C ∪
B)) ∩ C").pack()

    def do_next_step(self):
        if self.current_step >= len(self.steps_desc):
            messagebox.showinfo("Інформація", "Всі кроки виконано.");return

        desc = self.steps_desc[self.current_step]

        if desc.startswith("U ="):
            self.label_operation_result.config(text=f"U = {sorted(self.U)}")
        elif desc == "Ā = U \\ A":
            self.A_bar = self.U - self.A
            self.label_operation_result.config(text=f"Ā =
{sorted(self.A_bar)}")
        elif desc == "B̄ = U \\ B":
            self.B_bar = self.U - self.B
            self.label_operation_result.config(text=f"B̄ =
{sorted(self.B_bar)}")
        elif desc == "A∩B̄":
            self.step_AintBbar = self.A.intersection(self.B_bar)
            self.label_operation_result.config(text=f"A∩B̄ =
{sorted(self.step_AintBbar)}")
        elif desc == "B∩Ā":
            self.step_BintAbar = self.B.intersection(self.A_bar)
            self.label_operation_result.config(text=f"B∩Ā =
{sorted(self.step_BintAbar)}")
        elif desc == "(A∩B̄) ∪ (B∩Ā)":
            self.step_sym_diff_AB =
self.step_AintBbar.union(self.step_BintAbar)
            self.label_operation_result.config(text=f"(A∩B̄) ∪ (B∩Ā) =
{sorted(self.step_sym_diff_AB)}")
        elif desc == "C ∪ B":
            self.step_CuB = self.C.union(self.B)
            self.label_operation_result.config(text=f"C ∪ B =
{sorted(self.step_CuB)}")
        elif desc == "((A∩B̄) ∪ (B∩Ā)) ∩ (C ∪ B)":
            self.step_intermediate =
self.step_sym_diff_AB.intersection(self.step_CuB)
            self.label_operation_result.config(text=f"((A∩B̄) ∪ (B∩Ā)) ∩ (C
∪ B) = {sorted(self.step_intermediate)}")
        elif desc == "D = [((A∩B̄) ∪ (B∩Ā)) ∩ (C ∪ B)] ∩ C":
            self.D = self.step_intermediate.intersection(self.C)
            self.label_operation_result.config(text=f"D = {sorted(self.D)}")
            self.label_D.config(text=f"D: {sorted(self.D)}")
            self.button_save_D.config(state='normal')

        self.current_step += 1
        self.label_current_step.config(text=f"Крок
{self.current_step}/{len(self.steps_desc)}: {desc}")

    def save_D_to_file(self):
```

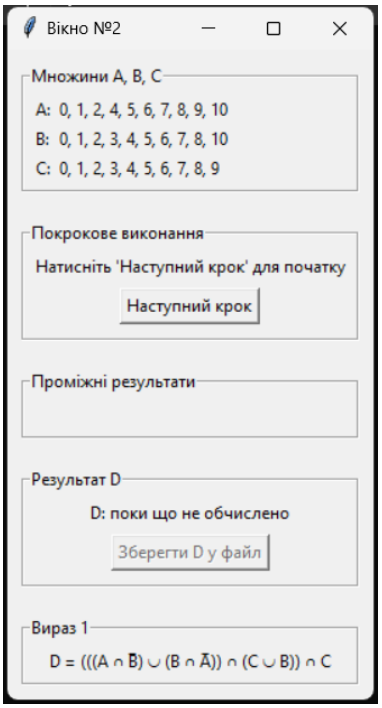
```
        if self.D is None:
            messagebox.showwarning("Увага", "D не обчислена."); return
        filename = filedialog.asksaveasfilename(defaultextension=".txt",
title="Зберегти D")
        if filename:
            with open(filename, "w") as f:
                f.write(", ".join(map(str, sorted(self.D))))
            messagebox.showinfo("Інформація", "Множина D збережена у файл.")

class Window3:

    def __init__(self, master, D, min_val, max_val):
        self.top = Toplevel(master)
        self.top.title("Вікно №3")

        self.D = D
        self.min_val = min_val
        self.max_val = max_val
        self.U = set(range(self.min_val, self.max_val+1))

        self.F = set()
        self.X = set()

        self.frame_D = tk.LabelFrame(self.top, text="Множина D", padx=5,
pady=5)
        self.frame_D.pack(padx=10, pady=10, fill='x')

        tk.Label(self.frame_D, text="D:").grid(row=0, column=0, sticky='w')
        self.label_D = tk.Label(self.frame_D, text=", ".join(map(str,
sorted(self.D))))
        self.label_D.grid(row=0, column=1, sticky='w')

        self.frame_F = tk.LabelFrame(self.top, text="Множина F", padx=5,
pady=5)
        self.frame_F.pack(padx=10, pady=10, fill='x')

        self.button_generate_F = tk.Button(self.frame_F, text="Згенерувати
F", command=self.generate_F)
        self.button_generate_F.pack(pady=5)

        self.label_F = tk.Label(self.frame_F, text="F: поки що не
згенеровано")
        self.label_F.pack()

        self.frame_X = tk.LabelFrame(self.top, text="Множина X", padx=5,
pady=5)
        self.frame_X.pack(padx=10, pady=10, fill='x')

        self.label_X = tk.Label(self.frame_X, text="X: поки що не обчислено")
        self.label_X.pack()

        self.button_calc_X = tk.Button(self.frame_X, text="Обчислити X",
command=self.calculate_X, state='disabled')
        self.button_calc_X.pack(pady=5)

        self.button_save_X = tk.Button(self.frame_X, text="Зберегти X у
```

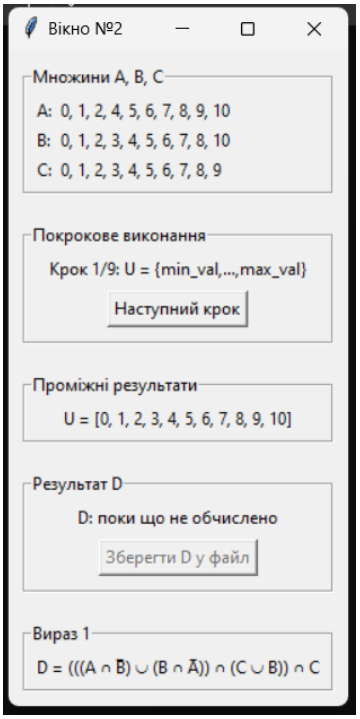
```
файл", command=self.save_X_to_file, state='disabled')
        self.button_save_X.pack(pady=5)

        self.frame_expr2 = tk.LabelFrame(self.top, text="Вираз 2", padx=5,
pady=5)
        self.frame_expr2.pack(padx=10, pady=10, fill='x')

        tk.Label(self.frame_expr2, text="X = F̄ ∪ D̄").pack()

    def generate_F(self):
        if not self.D:
            messagebox.showwarning("Увага", "D порожня.")
            return
        size = len(self.D)
        if size == 0:
            messagebox.showwarning("Увага", "D порожня, немає з чого
генерувати F.")
            return
        min_val = min(self.D)
        max_val = max(self.D)
        universe_sub = list(range(min_val, max_val+1))
        if size > len(universe_sub):
            messagebox.showerror("Помилка", "Неможливо згенерувати F з такими
умовами.")
            return

        chosen = set()
        chosen.add(min_val)
        chosen.add(max_val)
        while len(chosen) < size:
            chosen.add(random.choice(universe_sub))
        self.F = chosen
        self.label_F.config(text=f"F: {sorted(self.F)}")
        self.button_calc_X.config(state='normal')

    def calculate_X(self):
        if not self.F:
            messagebox.showwarning("Увага", "Спершу згенеруйте F.")
            return
        # X = F̄ ∪ D̄
        F_bar = self.U - self.F
        D_bar = self.U - self.D
        self.X = F_bar.union(D_bar)
        self.label_X.config(text=f"X: {sorted(self.X)}")
        self.button_save_X.config(state='normal')

    def save_X_to_file(self):
        if not self.X:
            messagebox.showwarning("Увага", "X порожня або не обчислена.")
            return
        filename = filedialog.asksaveasfilename(defaultextension=".txt",
title="Зберегти X")
        if filename:
            with open(filename, "w") as f:
                f.write(", ".join(map(str, sorted(self.X))))
            messagebox.showinfo("Інформація", "Множина X збережена у файл.")
```

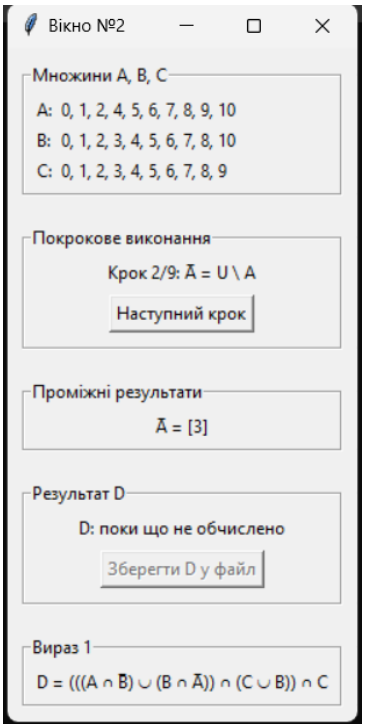
```
if __name__ == "__main__":
    root = tk.Tk()
    app = Window1(root)
    root.mainloop()
```

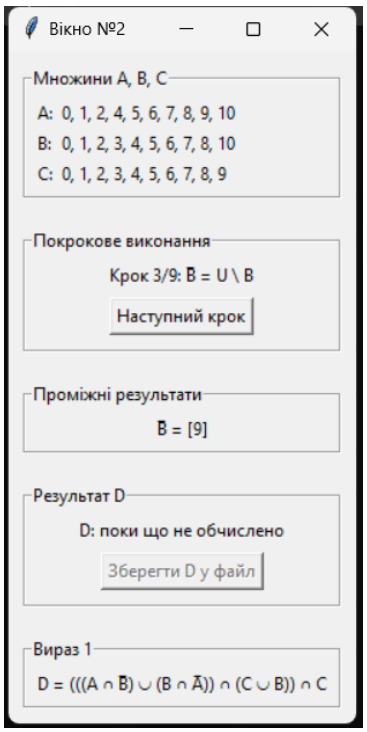
Знімки екрана контрольного прикладу:

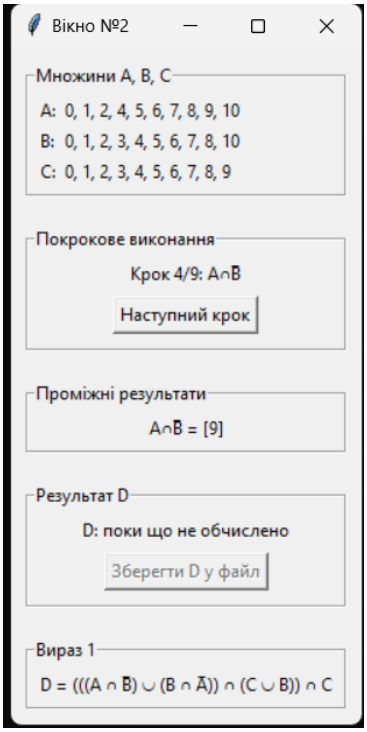


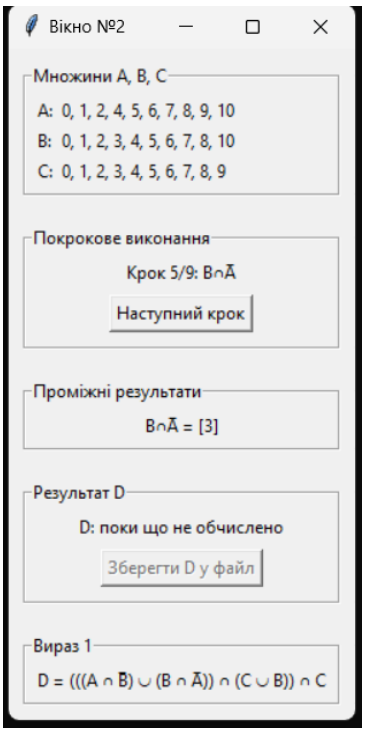


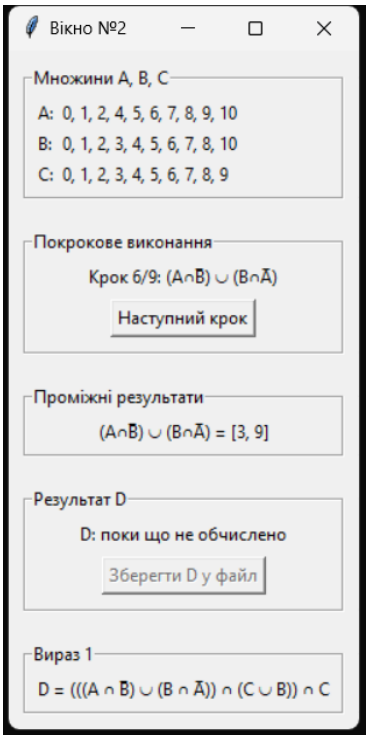


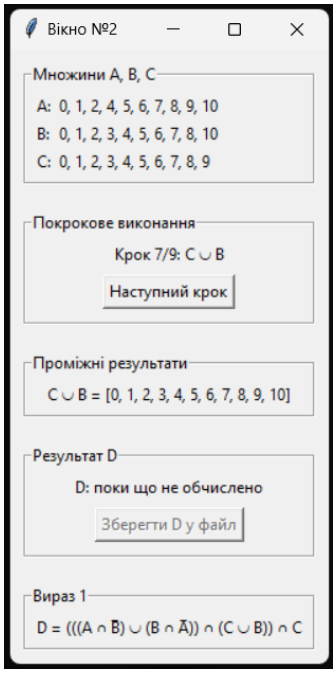


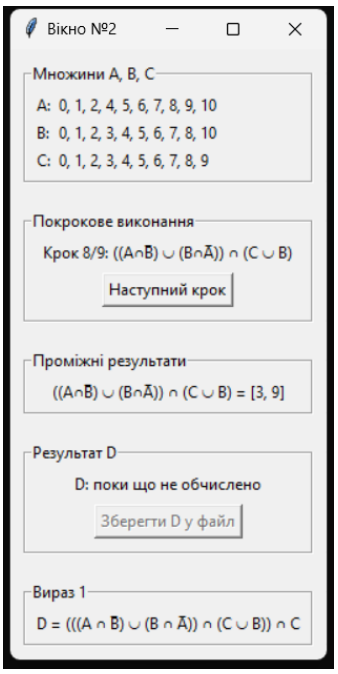


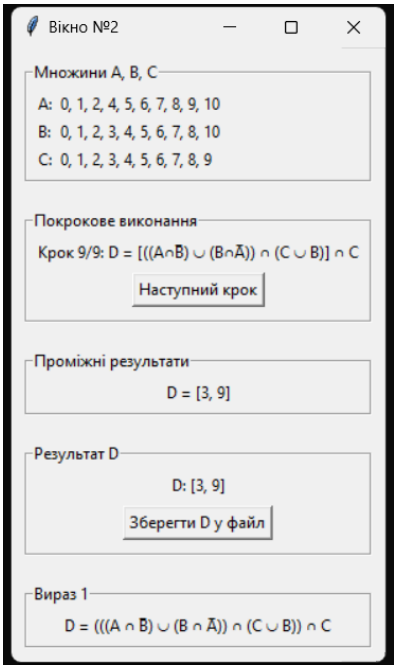


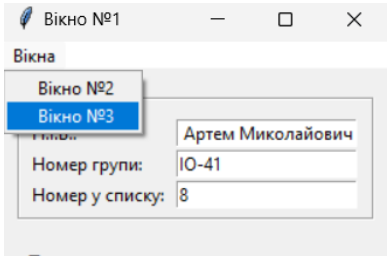


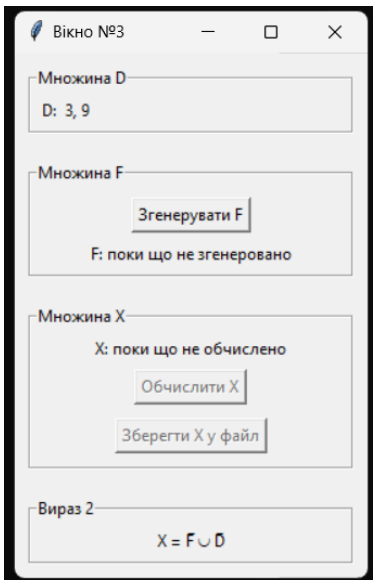


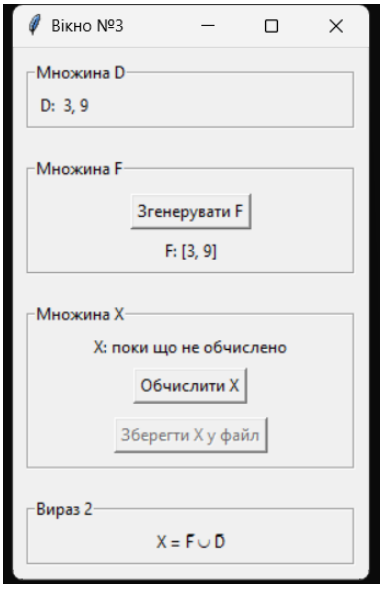


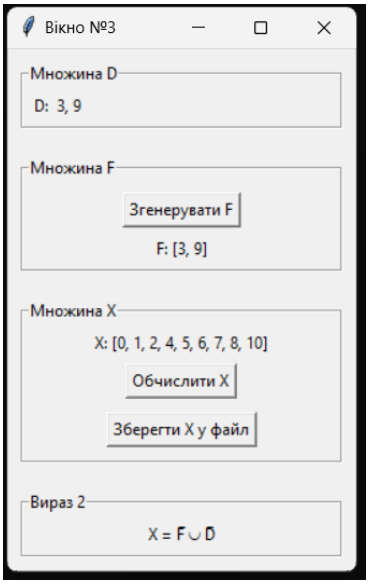












Висновок:

У ході виконання цієї лабораторної роботи я навчився працювати з графічним інтерфейсом у Python за допомогою модуля Tkinter. Я зрозумів, як створювати головне вікно, підвікна (Toplevel), меню та основні віджети (Label, Entry, Button, LabelFrame). Окрім того, я навчився виконувати логічні операції над множинами та відображати проміжні й кінцеві результати у віджетах інтерфейсу. Я реалізував покрокове обчислення виразів, можливість збереження результатів у файл та тестування роботи програми. Демонстрація коректності виконання роботи була проведена шляхом успішної взаємодії з інтерфейсом та перевірки одержаних результатів.