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

Факультет інформатики та обчислювальної техніки

Кафедра обчислювальної техніки

# Дискретна математика

# Лабораторна робота № 1

«Множини: основні властивості та операції над ними, діаграми Венна»

Виконав:
студент групи ІО-41
*Давидчук А. М.*
Залікова книжка № 4106

Перевірив
*Пономаренко А. М.*

Київ-2024

**Тема:** «Множини: основні властивості та операції над ними, діаграми Венна».

**Мета:** вивчити основні аксіоми, закони та теореми теорії множин, навчитися застосовувати їх на практиці. Обчислити логічний вираз шляхом послідовного застосування операцій над множинами.

**Загальне завдання:**

А) Вибрати номер Z індивідуального варіанта відповідно до виразу: Z=(i+G%60)%30+1, де i – номер у списку групи, G – числова частина назви групи.
Б) Максимально спростити логічний вираз. Для спрощення використати тотожності алгебри множин та визначення логічних операцій. Спрощення є максимальним, якщо формула містить тільки одне входження кожної множини.
В) Створити блок-схему послідовності обчислення початкового логічного виразу.
Г) Створити блок-схему послідовності обчислення спрощеного логічного виразу.
Д) З використанням блок-схем створити проєкт, який містить модуль з функцією для обчислення початкового виразу (1) та модуль з функцією для обчислення спрощеного виразу.
Е) З використанням блок-схем, заданих у лабораторній роботі, створити модуль з функцією для виконання логічної операції (2), вибраної відповідно до варіанта.
Ж) В основному файлі виконати перевірку правильності спрощення виразу з виводом відповідного повідомлення.
З) Як елементи множин можуть бути використані числа від 0 до 255.
І) Лабораторну роботу виконувати з застосуванням мови Python та бібліотеки tkinter.

**Короткі теоретичні відомості:**

*Властивості множин*
Множина – є сукупність визначених об'єктів, різних між собою, об’єднаних за певною ознакою чи властивістю. Множини позначають великими латинськими буквами. Скінченна множина – це така множина, кількість елементів якої може бути виражена скінченним числом, причому не важливо, чи можемо ми порахувати це число в цей момент. Потужність множини. Кількість елементів у скінченній множині A називають потужністю множини A і позначають A . Універсальна множина U є множина, що має таку властивість, що всі розглянуті множини є її підмножинами.

*Операції над множинами*
Об'єднанням множин A і B називають множину, що складається із всіх тих елементів, які належать хоча б одній з множин A або B.
Перетином множин A і B називають множину, що складається із всіх тих елементів, які належать як множині A, так і множині B.
Доповненням (або абсолютним доповненням) множини A називають множину, що складається із всіх елементів універсальної множини, які не належать A.
Різницею множин A й B (або відносним доповненням) називають множину, що складається із всіх елементів множини A, які не належать B.
Симетричною різницею множин A і B називають множину, що складається з об'єднання всіх елементів, що належать множині A і не містяться в B, і елементів, що належать

множині B і не містяться в A.

Операції, які виконують над однією множиною, називають унарними. Операції, які виконують над двома множинами, називають бінарними. Прикладом унарної операції є знаходження доповнення. Прикладами бінарних операцій є об'єднання, перетин, різниця, симетрична різниця.

| | |
|---|---|
| 1. Комутативність об'єднання<br>$A \cup B = B \cup A$ | 1. Комутативність перетину<br>$A \cap B = B \cap A$ |
| 2. Асоціативність об'єднання<br>$A \cup (B \cup C) = (A \cup B) \cup C$ | 2. Асоціативність перетину<br>$A \cap (B \cap C) = (A \cap B) \cap C$ |
| 3. Дистрибутивність об'єднання відносно перетину<br>$A \cup (B \cap C) = (A \cup B) \cap (A \cup C)$ | 3. Дистрибутивність перетину відносно об'єднання<br>$A \cap (B \cup C) = (A \cap B) \cup (A \cap C)$ |
| 4. Закони дій з пустою та універсальною множинами<br>$A \cup \varnothing = A$<br>$A \cup \overline{A} = U$<br>$A \cup U = U$ | 4. Закони дій з пустою та універсальною множинами<br>$A \cap U = A$<br>$A \cap \overline{A} = \varnothing$<br>$A \cap \varnothing = \varnothing$ |
| 5.Закон ідемпотентності об'єднання<br>Термін **ідемпотентність** означує властивість математичного об'єкта, яка проявляється в тому, що повторна дія над об'єктом *не змінює* його<br>$A \cup A = A$ | 5. Закон ідемпотентності перетину<br>$A \cap A = A$ |
| 6. Закон де Моргана<br>$\overline{A \cup B} = \overline{A} \cap \overline{B}$ | 6. Закон де Моргана<br>$\overline{A \cap B} = \overline{A} \cup \overline{B}$ |
| 7. Закон поглинання<br>$A \cup (A \cap B) = A$ | 7. Закон поглинання<br>$A \cap (A \cup B) = A$ |
| 8. Закон склеювання<br>$(A \cap B) \cup (A \cap \overline{B}) = A$ | 8. Закон склеювання<br>$(A \cup B) \cap (A \cup \overline{B}) = A$ |
| 9. Закон Порецького<br>$A \cup (\overline{A} \cap B) = A \cup B$ | 9. Закон Порецького<br>$A \cap (\overline{A} \cup B) = A \cap B$ |
| 10. Закон подвійного доповнення $\overline{\overline{A}} = A$ | |
| 11. Визначення операції «різниця»: $A \setminus B = A \cap \overline{B}$. | |
| 12. Визначення операції «симетрична різниця»: $A \Delta B = (A \setminus B) \cup (B \setminus A)$. | |

**Індивідуальне Завдання:**

За кодом мій варіант 18:

```
>>> G = 41
>>> N = 6
>>> (N + G%60)%30+1
18
```

Тобто такий логічний вираз:

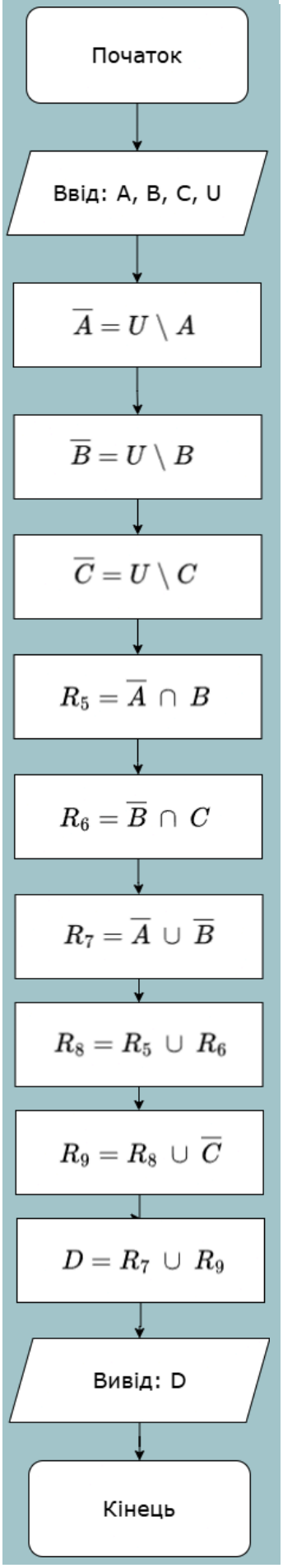
| | | |
|---|---|---|
| 18 | (1)<br>(2) | $D = \overline{A} \cup \overline{B} \cup \left(\overline{A} \cap B\right) \cup \left(\overline{B} \cap C\right) \cup \overline{C}$<br>$X = \overline{A}; Y = \overline{B}; Z = X \cap Y$ |

Спрощення логічного виразу $D$:

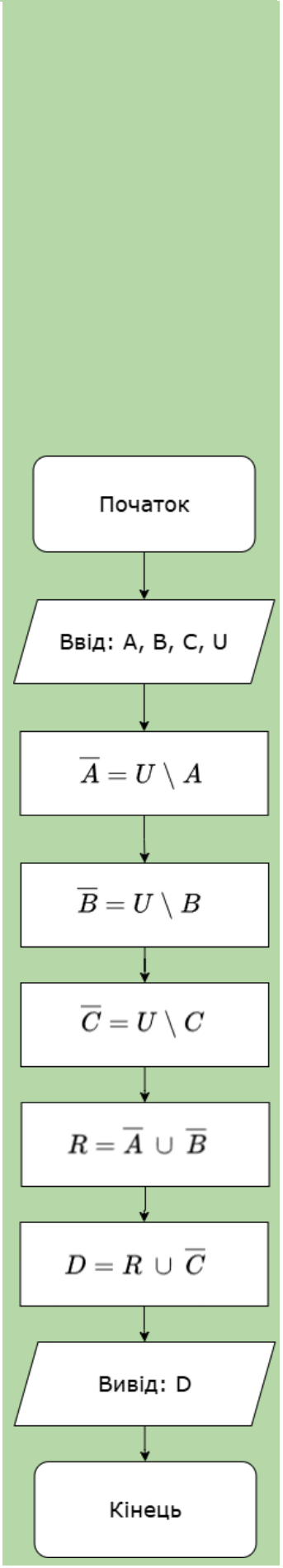
Використаю закон поглинання: $\overline{A} \cup (\overline{A} \cap B) = \overline{A}$ та $\overline{B} \cup (\overline{B} \cap C) = \overline{B}$. Виконаю заміну та отримаю наступний мінімізований логічний вираз: $D = \overline{A} \cup \overline{B} \cup \overline{C}$

Побудую блок-схеми послідовності обчислення початкового та мінімізованого логічних виразів:

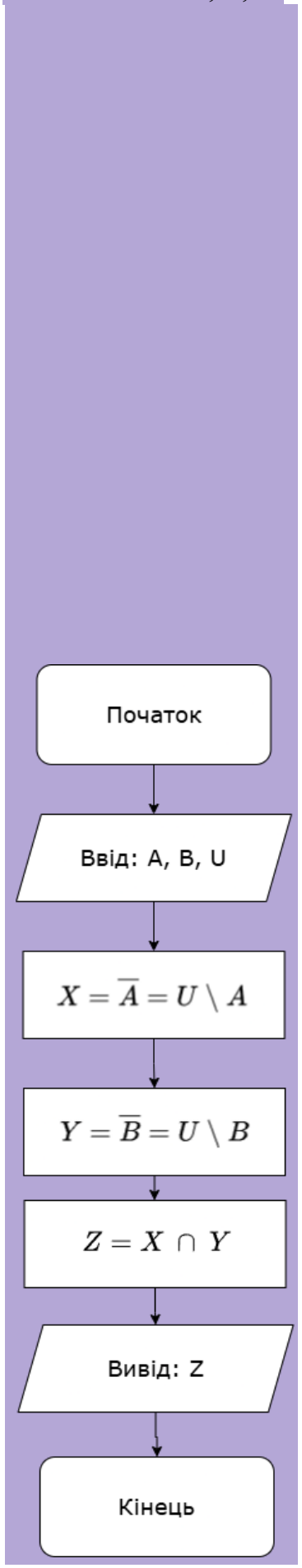
Основний D:




Мінімізований D:




X, Y, Z:



**Код:**

Файл *main.py*:

```
import tkinter as tk
from tkinter import messagebox
from random import sample

from init_expression import calculate_initial_expression
from mini_expression import calculate_minimized_expression
from final_expression import calculate_final_expression

class MainApp:

    def __init__(self, root: tk.Tk):

        # Налаштування головного вікна
        self.root = root
        self.root.title("Lab1")
        self.root.geometry("1200x600")
        self.root.resizable(False, False)
        self.root.configure(bg="#E2FFE2") # Зелений фон (на смак)

        self.norm_flag = False
        self.min_flag = False

        # 1. Верхній фрейм з інформацією про студента та результат перевірки
        info_frame = tk.Frame(self.root, bg="#E2FFE2", padx=10, pady=5)
        info_frame.pack(fill="x")

                tk.Label(info_frame,  text="П.І.Б:  Давидчук  Артем  Миколайович",
bg="#E2FFE2",
                font=("Arial", 12, "bold")).grid(row=0, column=0, sticky="w")
        tk.Label(info_frame, text="Група: ІО-41", bg="#E2FFE2",
                font=("Arial", 12)).grid(row=1, column=0, sticky="w")
        tk.Label(info_frame, text="Номер у списку: 6", bg="#E2FFE2",
                font=("Arial", 12)).grid(row=2, column=0, sticky="w")
        tk.Label(info_frame, text="Результат обчислення варіанту:", bg="#E2FFE2",
                font=("Arial", 12)).grid(row=3, column=0, sticky="w")

        self.variant_result = tk.StringVar(value="Невідомо")
                tk.Label(info_frame,  textvariable=self.variant_result,  fg="blue",
bg="#E2FFE2",
                font=("Arial", 12, "italic")).grid(row=3, column=1, sticky="w")

        # 2. Фрейм для параметрів множин
        param_frame = tk.Frame(self.root, bg="#E2FFE2", padx=10, pady=5)
        param_frame.pack(fill="x")

        tk.Label(param_frame, text="Потужність множини A:", bg="#E2FFE2",
                font=("Arial", 11)).grid(row=0, column=0, sticky="w")
        tk.Label(param_frame, text="Потужність множини B:", bg="#E2FFE2",
                font=("Arial", 11)).grid(row=1, column=0, sticky="w")
        tk.Label(param_frame, text="Потужність множини C:", bg="#E2FFE2",
                font=("Arial", 11)).grid(row=2, column=0, sticky="w")

        self.power_A = tk.IntVar(value=10)
        self.power_B = tk.IntVar(value=10)
        self.power_C = tk.IntVar(value=10)

             tk.Entry(param_frame,  textvariable=self.power_A,  width=5).grid(row=0,
```

```
column=1, padx=5)
            tk.Entry(param_frame, textvariable=self.power_B, width=5).grid(row=1,
column=1, padx=5)
            tk.Entry(param_frame, textvariable=self.power_C, width=5).grid(row=2,
column=1, padx=5)

        tk.Label(param_frame, text="Діапазон U (початок, кінець):", bg="#E2FFE2",
                 font=("Arial", 11)).grid(row=3, column=0, sticky="w")

        self.u_start = tk.IntVar(value=0)
        self.u_end = tk.IntVar(value=255)
            tk.Entry(param_frame, textvariable=self.u_start, width=5).grid(row=3,
column=1, padx=5)
             tk.Entry(param_frame, textvariable=self.u_end, width=5).grid(row=3,
column=2, padx=5)

        # 3. Фрейм з кнопками генерації та перевірки
        action_frame = tk.Frame(self.root, bg="#E2FFE2", padx=10, pady=5)
        action_frame.pack(fill="x")

        tk.Button(action_frame, text="Згенерувати множини A, B, C",
                        command=self.generate_sets, bg="#C8FFC8").pack(side="left",
padx=5)
        tk.Button(action_frame, text="Перевірити спрощення виразу",
                     command=self.check_expression, bg="#C8FFC8").pack(side="left",
padx=5)

        # 4. Фрейм для ручного вводу множин A, B, C
        manual_frame = tk.Frame(self.root, bg="#E2FFE2", padx=10, pady=5)
        manual_frame.pack(fill="x")

        tk.Label(manual_frame, text="Ручний ввід множин A, B, C (через кому):",
                 bg="#E2FFE2", font=("Arial", 11, "underline"))\
                 .grid(row=0, column=0, columnspan=3, sticky="w", pady=5)

            tk.Label(manual_frame, text="A:", bg="#E2FFE2").grid(row=1, column=0,
sticky="e")
        self.entry_A_manual = tk.Entry(manual_frame, width=40)
        self.entry_A_manual.grid(row=1, column=1, padx=5)

            tk.Label(manual_frame, text="B:", bg="#E2FFE2").grid(row=2, column=0,
sticky="e")
        self.entry_B_manual = tk.Entry(manual_frame, width=40)
        self.entry_B_manual.grid(row=2, column=1, padx=5)

            tk.Label(manual_frame, text="C:", bg="#E2FFE2").grid(row=3, column=0,
sticky="e")
        self.entry_C_manual = tk.Entry(manual_frame, width=40)
        self.entry_C_manual.grid(row=3, column=1, padx=5)

        tk.Button(manual_frame, text="Задати множини", bg="#C8FFC8",
                  command=self.set_manual_sets).grid(row=4, column=1, pady=5)

        # Ініціалізація множин
        self.set_A = set()
        self.set_B = set()
        self.set_C = set()
        self.universal_set = set()

        # 5. Фрейм з кнопками для виклику інших вікон
        menu_frame = tk.Frame(self.root, bg="#E2FFE2", padx=10, pady=10)
        menu_frame.pack()
```

```
        tk.Button(menu_frame, text="Вікно 2", width=20, bg="#FFE2E2",
                  command=self.open_window2).pack(side="left", padx=5)
        tk.Button(menu_frame, text="Вікно 3", width=20, bg="#FFE2E2",
                  command=self.open_window3).pack(side="left", padx=5)
        tk.Button(menu_frame, text="Вікно 4", width=20, bg="#FFE2E2",
                  command=self.open_window4).pack(side="left", padx=5)
        tk.Button(menu_frame, text="Вікно 5", width=20, bg="#FFE2E2",
                  command=self.open_window5).pack(side="left", padx=5)

    # Генерація множин
    def generate_sets(self):

          # Генерує випадкові множини A, B, C згідно з заданими потужностями та універсальною множиною

        try:
            start = self.u_start.get()
            end = self.u_end.get()

                if start > end: raise ValueError("Невірний діапазон універсальної множини!")
              if start < 0 or end > 255: raise ValueError("Діапазон U має бути [0; 255]")

            self.universal_set = set(range(start, end + 1))

            if (len(self.universal_set) < self.power_A.get() or
                len(self.universal_set) < self.power_B.get() or
                len(self.universal_set) < self.power_C.get()):
                    messagebox.showerror("Помилка", "Діапазон U замалий для заданої потужності!"); return

            # Генеруємо
            self.set_A = set(sample(list(self.universal_set), self.power_A.get()))
            self.set_B = set(sample(list(self.universal_set), self.power_B.get()))
            self.set_C = set(sample(list(self.universal_set), self.power_C.get()))

            msg = f"Множини згенеровано:\nA = {self.format_set(self.set_A)}\n" \
                  f"B = {self.format_set(self.set_B)}\n" \
                  f"C = {self.format_set(self.set_C)}"
            messagebox.showinfo("Інформація", msg)

        except Exception as e: messagebox.showerror("Помилка", str(e))

    def check_expression(self):

             # Перевіряє правильність спрощення виразу, порівнюючи початковий та мінімізований варіанти

        try:
            # Обчислюємо початковий та мінімізований
              result_initial = calculate_initial_expression(self.set_A, self.set_B, self.set_C, self.universal_set)
                    result_minimized = calculate_minimized_expression(self.set_A, self.set_B, self.set_C, self.universal_set)

                                       if result_initial == result_minimized: self.variant_result.set("Спрощення виразу виконано правильно")
            else: self.variant_result.set("Помилка в спрощенні виразу")

        except Exception as e: messagebox.showerror("Помилка", str(e))
```

```
    # Ручне введення множин A, B, C
    def set_manual_sets(self):

        # Зчитує рядки з entry_A_manual, entry_B_manual, entry_C_manual, перетворює на цілі числа і зберігає у self.set_A, self.set_B, self.set_C.

        start = self.u_start.get()
        end = self.u_end.get()

             if start > end: raise ValueError("Невірний діапазон універсальної множини!")
        if start < 0 or end > 255: raise ValueError("Діапазон U має бути [0; 255]")

        self.universal_set = set(range(start, end + 1))

        try:
            # A
            a_str = self.entry_A_manual.get().strip()
            if a_str:
                a_list = [int(x.strip()) for x in a_str.split(',') if x.strip()]
                self.set_A = set(a_list)
            else:
                self.set_A = set()

            if self.set_A and (max(self.set_A) > 255 or min(self.set_A) < 0): raise ValueError("Помилка", "Числа мають бути цілими числами від 0 до 255.")

            # B
            b_str = self.entry_B_manual.get().strip()
            if b_str:
                b_list = [int(x.strip()) for x in b_str.split(',') if x.strip()]
                self.set_B = set(b_list)
            else:
                self.set_B = set()

            if self.set_B and (max(self.set_B) > 255 or min(self.set_B) < 0): raise ValueError("Помилка", "Числа мають бути цілими числами від 0 до 255.")

            # C
            c_str = self.entry_C_manual.get().strip()
            if c_str:
                c_list = [int(x.strip()) for x in c_str.split(',') if x.strip()]
                self.set_C = set(c_list)
            else:
                self.set_C = set()

            if self.set_C and (max(self.set_C) > 255 or min(self.set_C) < 0): raise ValueError("Помилка", "Числа мають бути цілими числами від 0 до 255.")

            msg = f"Задані вручну множини:\nA = {self.format_set(self.set_A)}\n" \
                  f"B = {self.format_set(self.set_B)}\n" \
                  f"C = {self.format_set(self.set_C)}"
            messagebox.showinfo("Ручний ввід", msg)

        except ValueError:
              messagebox.showerror("Помилка", "Некоректний формат введення. Вводьте цілі числа через кому та в діапазоні [0; 255].")
        except Exception as e:
            messagebox.showerror("Помилка", str(e))

    # Вікно 2 (Покрокове виконання початкового виразу)
```

```
    def open_window2(self):

        win2 = tk.Toplevel(self.root)
        win2.title("Вікно 2 - Початковий вираз")
        win2.resizable(False, False)
        win2.geometry("900x500")
        win2.configure(bg="#E2F2FF")

        # Відображення множин A, B, C
        tk.Label(win2, text=f"A = {self.format_set(self.set_A)}", bg="#E2F2FF",
                 font=("Arial", 11)).pack(anchor="w", padx=10, pady=2)
        tk.Label(win2, text=f"B = {self.format_set(self.set_B)}", bg="#E2F2FF",
                 font=("Arial", 11)).pack(anchor="w", padx=10, pady=2)
        tk.Label(win2, text=f"C = {self.format_set(self.set_C)}", bg="#E2F2FF",
                 font=("Arial", 11)).pack(anchor="w", padx=10, pady=2)

        # Текстове поле для покрокового виводу
        self.log_text2 = tk.Text(win2, width=80, height=15, state="disabled")
        self.log_text2.pack(padx=10, pady=5)

        # Кнопка "Наступний крок"
        tk.Button(win2, text="Наступний крок", bg="#FFFFFF",
                  command=self.next_step_initial).pack(pady=5)

        # Віджет для кінцевого результату
        self.result_D_label = tk.Label(win2, text="Результат D: (ще не обчислено)",
                                       bg="#E2F2FF", font=("Arial", 11, "bold"))
        self.result_D_label.pack(pady=5)

        tk.Button(win2, text="Зберегти результат у файл", bg="#FFFFCC",
                                                                command=lambda:
self.save_result_to_file(self.log_text2.get("1.0", tk.END),
                                                            "initial_result.txt",
type="norm")).pack(pady=5)

        # Налаштування покрокової логіки
        self.current_step = 0
        self.step_results = {}  # Словник для проміжних результатів

        # Приклад покрокових операцій для "початкового виразу"
        # Тут ви пишете реальну логіку: (ОПИС, ФУНКЦІЯ)
        self.initial_steps = [
                                      ("Крок   1:   !A   =   U   \\   A",   lambda:
self.universal_set.difference(self.set_A)),
                                      ("Крок   2:   !B   =   U   \\   B",   lambda:
self.universal_set.difference(self.set_B)),
                                      ("Крок   3:   !C   =   U   \\   C",   lambda:
self.universal_set.difference(self.set_C)),
                                      ("Крок   4:   R5   =   !A   ∩   B",   lambda:
self.step_results["!A"].intersection(self.set_B)),
                                      ("Крок   5:   R6   =   !B   ∩   C",   lambda:
self.step_results["!B"].intersection(self.set_C)),
                                     ("Крок   6:   R7   =   !A   ∪   !B",   lambda:
self.step_results["!A"].union(self.step_results["!B"])),
                                     ("Крок   7:   R8   =   R5   ∪   R6",   lambda:
self.step_results["R5"].union(self.step_results["R6"])),
                                     ("Крок   8:   R9   =   R8   ∪   !C",   lambda:
self.step_results["R8"].union(self.step_results["!C"])),
                                     ("Крок   9:   D   =   R7   ∪   R9",   lambda:
self.step_results["R7"].union(self.step_results["R9"])),
        ]
```

```
    def next_step_initial(self):

        # Виконує наступний крок з initial_steps і виводить у log_text2

         if not hasattr(self, "initial_steps"): return # Якщо вікно не відкрите або
не ініціалізоване
        if self.current_step >= len(self.initial_steps): return

        description, func = self.initial_steps[self.current_step]

        # Обчислюємо
        result = func()

        # Зберігаємо проміжний результат у self.step_results
        if "!A" in description[8:10]: self.step_results["!A"] = result
        elif "!B" in description[8:10]: self.step_results["!B"] = result
        elif "!C" in description[8:10]: self.step_results["!C"] = result
        elif "R5" in description[8:10]: self.step_results["R5"] = result
        elif "R6" in description[8:10]: self.step_results["R6"] = result
        elif "R7" in description[8:10]: self.step_results["R7"] = result
        elif "R8" in description[8:10]: self.step_results["R8"] = result
        elif "R9" in description[8:10]: self.step_results["R9"] = result
        elif "D" in description[8:10]: self.step_results["D"] = result

        # Логуємо
        self.log_text2.config(state='normal')
                              self.log_text2.insert(tk.END,    f"{description}    =>
{self.format_set(result)}\n\n")
        self.log_text2.config(state='disabled')

        # Якщо це останній крок (отримали D)
        if "D" in description:
            self.norm_flag = True
            self.D_norm = self.format_set(result)
                                   self.result_D_label.config(text=f"Результат   D:
{self.format_set(result)}")

        self.current_step += 1

    # Вікно 3 (Покрокове виконання мінімізованого виразу)
    def open_window3(self):

        win3 = tk.Toplevel(self.root)
        win3.title("Вікно 3 - Мінімізований вираз")
        win3.geometry("900x500")
        win3.resizable(False, False)
        win3.configure(bg="#F2E2FF")

        tk.Label(win3, text=f"A = {self.format_set(self.set_A)}", bg="#F2E2FF",
                 font=("Arial", 11)).pack(anchor="w", padx=10, pady=2)
        tk.Label(win3, text=f"B = {self.format_set(self.set_B)}", bg="#F2E2FF",
                 font=("Arial", 11)).pack(anchor="w", padx=10, pady=2)
        tk.Label(win3, text=f"C = {self.format_set(self.set_C)}", bg="#F2E2FF",
                 font=("Arial", 11)).pack(anchor="w", padx=10, pady=2)

        self.log_text3 = tk.Text(win3, width=80, height=15, state="disabled")
        self.log_text3.pack(padx=10, pady=5)

        tk.Button(win3, text="Наступний крок", bg="#FFFFFF",
                  command=self.next_step_minimized).pack(pady=5)

              self.result_D_min_label  =  tk.Label(win3,  text="Результат  D:  (ще  не
```

```
обчислено)",
                                                    bg="#F2E2FF", font=("Arial", 11,
"bold"))
        self.result_D_min_label.pack(pady=5)

        tk.Button(win3, text="Зберегти результат у файл", bg="#FFFFCC",
                                                              command=lambda:
self.save_result_to_file(self.log_text3.get("1.0", tk.END),
                                                          "minimized_result.txt",
type="min")).pack(pady=5)

        # Покрокова логіка для мінімізованого виразу
        self.current_step_min = 0
        self.min_steps = []
        self.min_step_results = {}

        self.min_steps = [
                                    ("Крок   1:   !A   =   U   \\   A",   lambda:
self.universal_set.difference(self.set_A)),
                                    ("Крок   2:   !B   =   U   \\   B",   lambda:
self.universal_set.difference(self.set_B)),
                                    ("Крок   3:   !C   =   U   \\   C",   lambda:
self.universal_set.difference(self.set_C)),
                                   ("Крок   4:   R   =   !A   ∪   !B",   lambda:
self.min_step_results["!A"].union(self.min_step_results["!B"])),
                                    ("Крок   5:   D   =   R   ∪   !C",   lambda:
self.min_step_results["R"].union(self.min_step_results["!C"])),
        ]

    def next_step_minimized(self):

        # Виконує наступний крок з min_steps і виводить у log_text3.

        if not hasattr(self, "min_steps"): return
        if self.current_step_min >= len(self.min_steps): return

        description, func = self.min_steps[self.current_step_min]
        result = func()

        if "!A" in description[8:10]: self.min_step_results["!A"] = result
        elif "!B" in description[8:10]: self.min_step_results["!B"] = result
        elif "!C" in description[8:10]: self.min_step_results["!C"] = result
        elif "R" in description[8:10]: self.min_step_results["R"] = result
        elif "D" in description[8:10]: self.min_step_results["D"] = result

        self.log_text3.config(state='normal')
                            self.log_text3.insert(tk.END,    f"{description}    =>
{self.format_set(result)}\n\n")
        self.log_text3.config(state='disabled')

        if "D" in description:
            self.min_flag = True
            self.D_min = self.format_set(result)
                              self.result_D_min_label.config(text=f"Результат   D:
{self.format_set(result)}")
        self.current_step_min += 1

    # Вікно 4 (Операція (2) над X і Y)
    def open_window4(self):

        win4 = tk.Toplevel(self.root)
        win4.title("Вікно 4 - Операція над множинами X і Y")
```

```
        win4.geometry("600x400")
        win4.resizable(False, False)
        win4.configure(bg="#FFE2F2")

                    result  =  calculate_final_expression(self.set_A,  self.set_B,
self.universal_set)
        Z = result[2]

        # Створюємо текстовий віджет із забороною редагування (state='disabled')
              self.log_text4  =  tk.Text(win4,  width=60,  height=15,  bg="#FFE2F2",
state='disabled')
        self.log_text4.pack(padx=10, pady=10)

        # Щоби щось вставити у Text, тимчасово робимо state='normal'
        self.log_text4.config(state='normal')
                    self.log_text4.insert(tk.END,  f"X  =  !A  =  U  \\  A  =>
{self.format_set(result[0])}\n\n")
                    self.log_text4.insert(tk.END,  f"Y  =  !B  =  U  \\  B  =>
{self.format_set(result[1])}\n\n\n")
        self.log_text4.insert(tk.END, f"Z = X ∩ Y => {self.format_set(Z)}\n")

        # Повторно забороняємо редагування
        self.log_text4.config(state='disabled')

        # Кнопка для збереження Z
        self.Z_norm = self.format_set(Z)
           tk.Button(win4, text="Зберегти Z у файл", bg="#FFFFFF", command=lambda:
self.save_result_to_file(self.format_set(Z) + "   ", "Z.txt", "Z")).pack(pady=5)

    #  Вікно 5 (Зчитування та порівняння результатів)
    def open_window5(self):

        win5 = tk.Toplevel(self.root)
        win5.title("Вікно 5 - Зчитування та порівняння")
        win5.geometry("900x400")
        win5.resizable(False, False)
        win5.configure(bg="#FFF2E2")

                # Текстовий віджет для всіх виводів, забороняємо редагування
(state='disabled')
        self.text_log5 = tk.Text(win5, width=100, height=15, state='disabled')
        self.text_log5.pack(padx=10, pady=10)

        # Змінні для збереження зчитаних/обчислених рядків
        self.d_initial_str = None      # Результат D(початкового)
        self.d_minimized_str = None    # Результат D(спрощеного)
        self.z_func_str = None         # Результат Z(функція)
        self.z_standard_str = None     # Результат Z(стандарт.)

        # Фрейм з кнопками внизу
        frame_btns = tk.Frame(win5, bg="#FFF2E2")
        frame_btns.pack(fill='x', pady=5)

        # Кнопки зчитування
        tk.Button(frame_btns, text="Зчитати D(поч.)",
                  command=lambda: self.read_and_show_set("initial_result.txt",
"D(поч.)"),
                bg="#FFFFFF").pack(side='left', padx=5)

        tk.Button(frame_btns, text="Зчитати D(спр.)",
                  command=lambda: self.read_and_show_set("minimized_result.txt",
"D(спр.)"),
```

```
                bg="#FFFFFF").pack(side='left', padx=5)

        tk.Button(frame_btns, text="Зчитати Z(функ.)",
                command=lambda: self.read_and_show_set("Z.txt", "Z(функ.)"),
                bg="#FFFFFF").pack(side='left', padx=5)

        # Кнопка обчислення "стандартної" операції
        tk.Button(frame_btns, text="Обчислити Z(стандарт.)",
                command=self.compute_standard_operation_in_window5,
                bg="#FFFFFF").pack(side='left', padx=5)

        # Кнопка порівняння результатів
        tk.Button(frame_btns, text="Порівняти результати",
                command=self.compare_results_in_window5,
                bg="#FFFFCC").pack(side='left', padx=5)

    def read_and_show_set(self, filename: str, label_name: str):

         # Зчитує вміст файлу 'filename', зберігає в одну зі змінних класу (залежно від файлу) і виводить у текстове поле self.text_log5

        try:

            with open(filename, "r", encoding="utf-8") as f: pass

            # Залежно від файлу зберігаємо у потрібну змінну
            if filename == "initial_result.txt":
                content = self.D_norm
                self.d_initial_str = self.D_norm
            elif filename == "minimized_result.txt":
                content = self.D_min
                self.d_minimized_str = content
            elif filename == "Z.txt":
                content = self.Z_norm
                self.z_func_str = content

            # Виводимо у текстовий віджет
            self.text_log5.config(state='normal')
            self.text_log5.insert(tk.END, f"{label_name}: {content}\n\n")
            self.text_log5.config(state='disabled')

              except FileNotFoundError: tk.messagebox.showerror("Помилка", f"Файл {filename} не знайдено.")
        except Exception as e: tk.messagebox.showerror("Помилка", str(e))

    def compute_standard_operation_in_window5(self):

             # Виконує логічну операцію над X і Y стандартними засобами Python (наприклад, об'єднання)

        try:

            X = self.universal_set.difference(self.set_A)
            Y = self.universal_set.difference(self.set_B)
            Z_standard = X.intersection(Y)

            # Перетворимо в рядок
            z_str = self.format_set(Z_standard)
            self.z_standard_str = z_str

            # Виводимо у текстове поле
            self.text_log5.config(state='normal')
```

```
            self.text_log5.insert(tk.END, f"Z(стандарт.): {z_str}\n")
            self.text_log5.config(state='disabled')

        except Exception as e: tk.messagebox.showerror("Помилка", str(e))

    def compare_results_in_window5(self):
            # Порівнює результати: (початковий vs. мінімізований) та (операція за функцією vs. стандартна операція).

        try:

            msg = ""
               if self.d_initial_str == self.d_minimized_str: msg += "Початковий і мінімізований результати D співпадають.\n"
                    else: msg += "Початковий і мінімізований результати D НЕ співпадають.\n"

             if self.z_func_str == self.z_standard_str: msg += "Результати операції Z(функція) і Z(стандартна) співпадають."
                  else: msg += "Результати операції Z(функція) і Z(стандартна) НЕ співпадають."

            messagebox.showinfo("Порівняння результатів", msg)

           except FileNotFoundError: messagebox.showerror("Помилка", "Деякі файли відсутні. Спочатку обчисліть і збережіть усі результати.")
        except Exception as e: messagebox.showerror("Помилка", str(e))

    #  Допоміжні методи: збереження, зчитування, порівняння
    def save_result_to_file(self, content, filename, type=None):

        if (type == "norm" and self.norm_flag) or (type == "min" and self.min_flag) or type == "Z":

            try:
                with open(filename, "w", encoding="utf-8") as f:
                    f.write(content[:-3])
                  messagebox.showinfo("Інформація", f"Результат збережено у файлі: {filename}")
            except Exception as e:
                messagebox.showerror("Помилка", str(e))

        else:

              if not self.norm_flag: messagebox.showerror("Помилка", "Не знайдено результату D (початкового)!")
              elif not self.min_flag: messagebox.showerror("Помилка", "Не знайдено результату D (спрощеного)!")

    #  Форматування множини (щоб замінити 'set()' на '{}')
    def format_set(self, s: set) -> str:
        """Виводить множину у вигляді {1, 2, 3}, якщо порожня - {}."""
        if not s: return "{}"
        return "{" + ", ".join(str(x) for x in sorted(s)) + "}"

root = tk.Tk()
app = MainApp(root)
root.mainloop()
```

Файл *init_expression.py*:

```
def calculate_initial_expression(A: set, B: set, C: set, U: set) -> set:

    not_A = U.difference(A)
    not_B = U.difference(B)
    not_C = U.difference(C)

    R5 = not_A.intersection(B)
    R6 = not_B.intersection(C)
    R7 = not_A.union(not_B)
    R8 = R5.union(R6)
    R9 = R8.union(not_C)
    D = R7.union(R9)

    return D
```

Файл *mini_expression.py*:

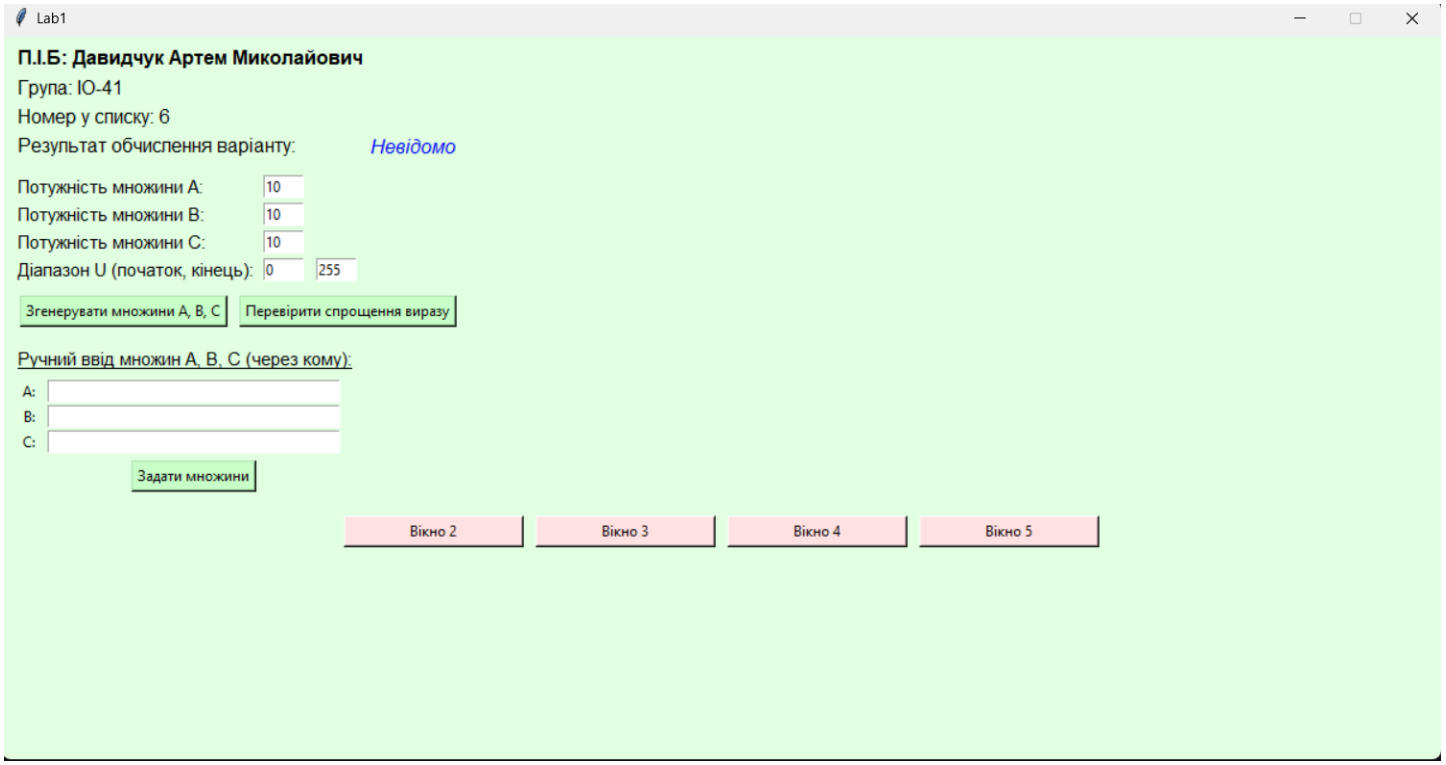
```
def calculate_minimized_expression(A: set, B: set, C: set, U: set) -> set:

    not_A = U.difference(A)
    not_B = U.difference(B)
    not_C = U.difference(C)

    R = not_A.union(not_B)
    D = R.union(not_C)

    return D
```

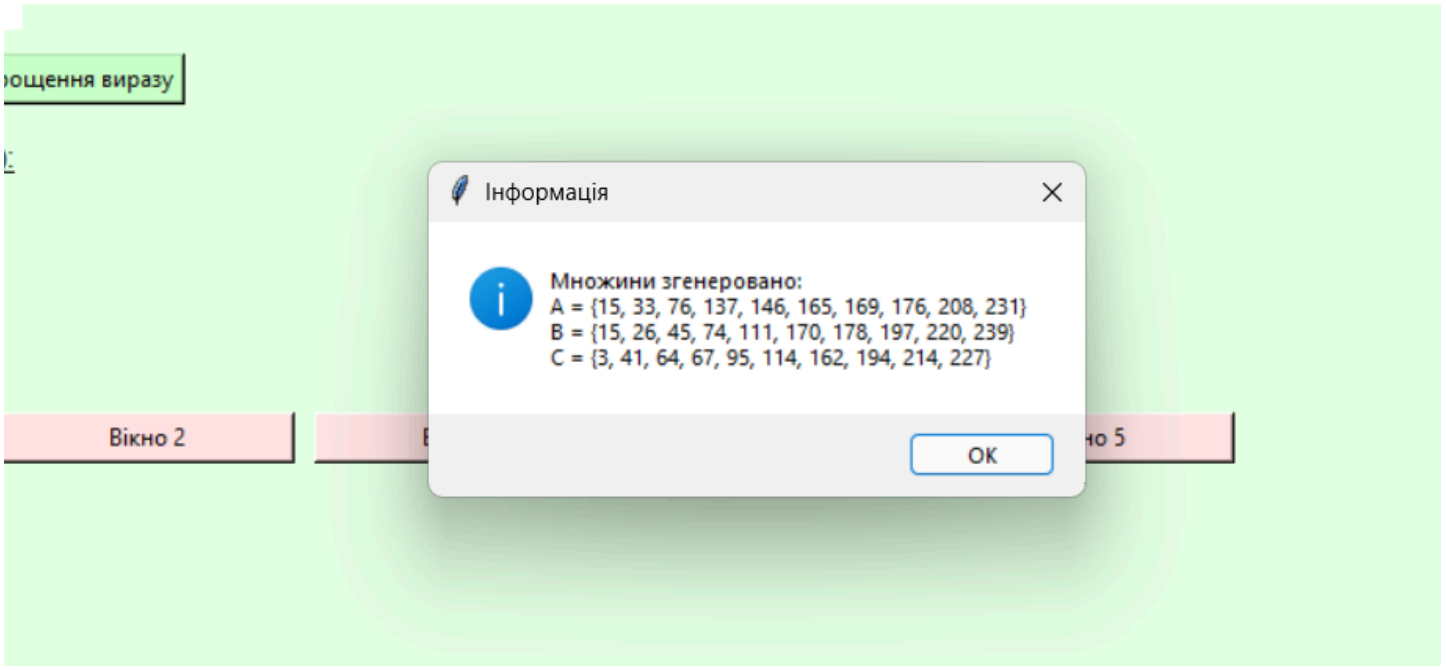
Файл *final_expression.py*:

```
def calculate_final_expression(A: set, B: set, U: set) -> set:

    X = U.difference(A)
    Y = U.difference(B)

    Z = X.intersection(Y)

    return X, Y, Z
```

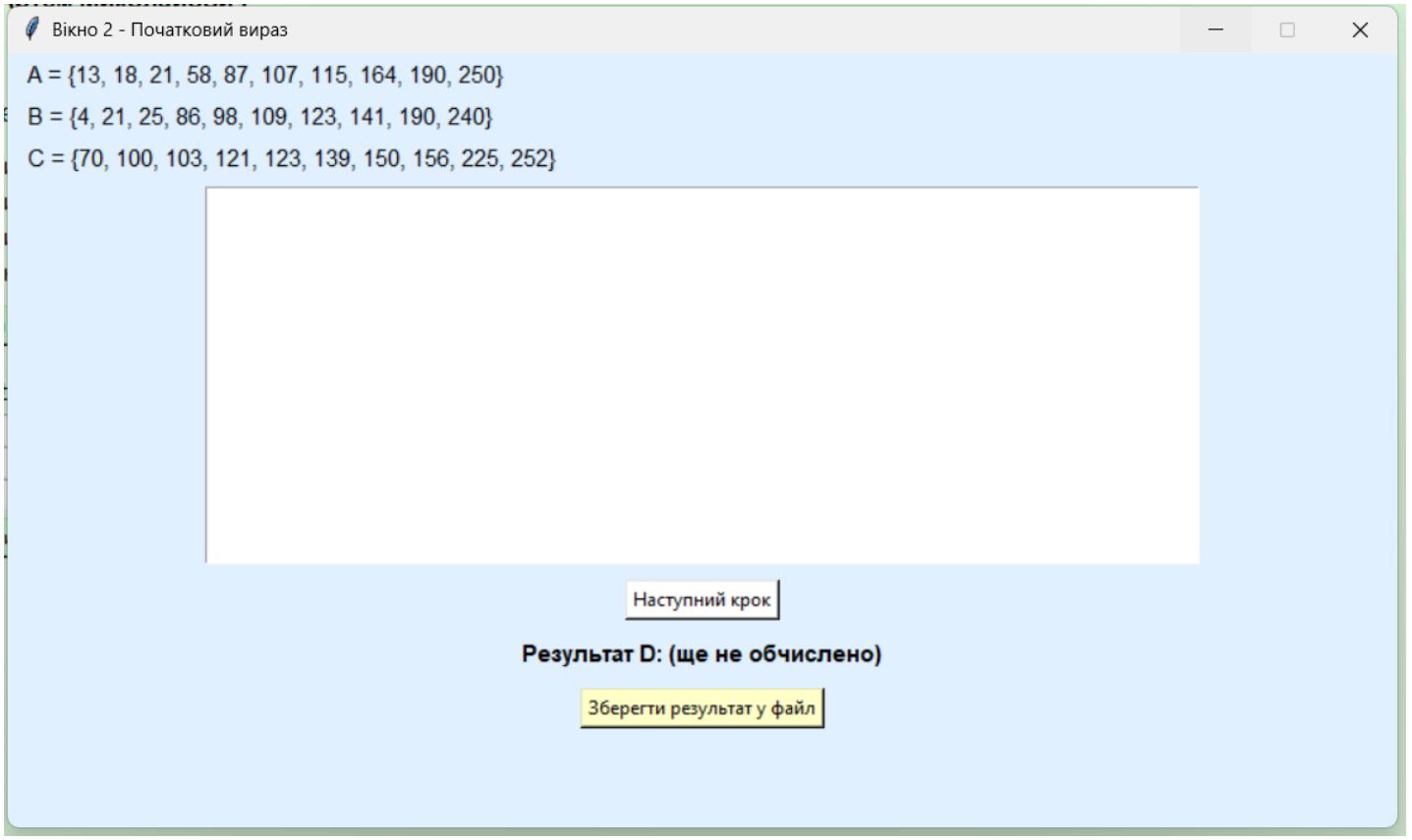
## Результати виконання (знімки екрана):

Вікно №1:

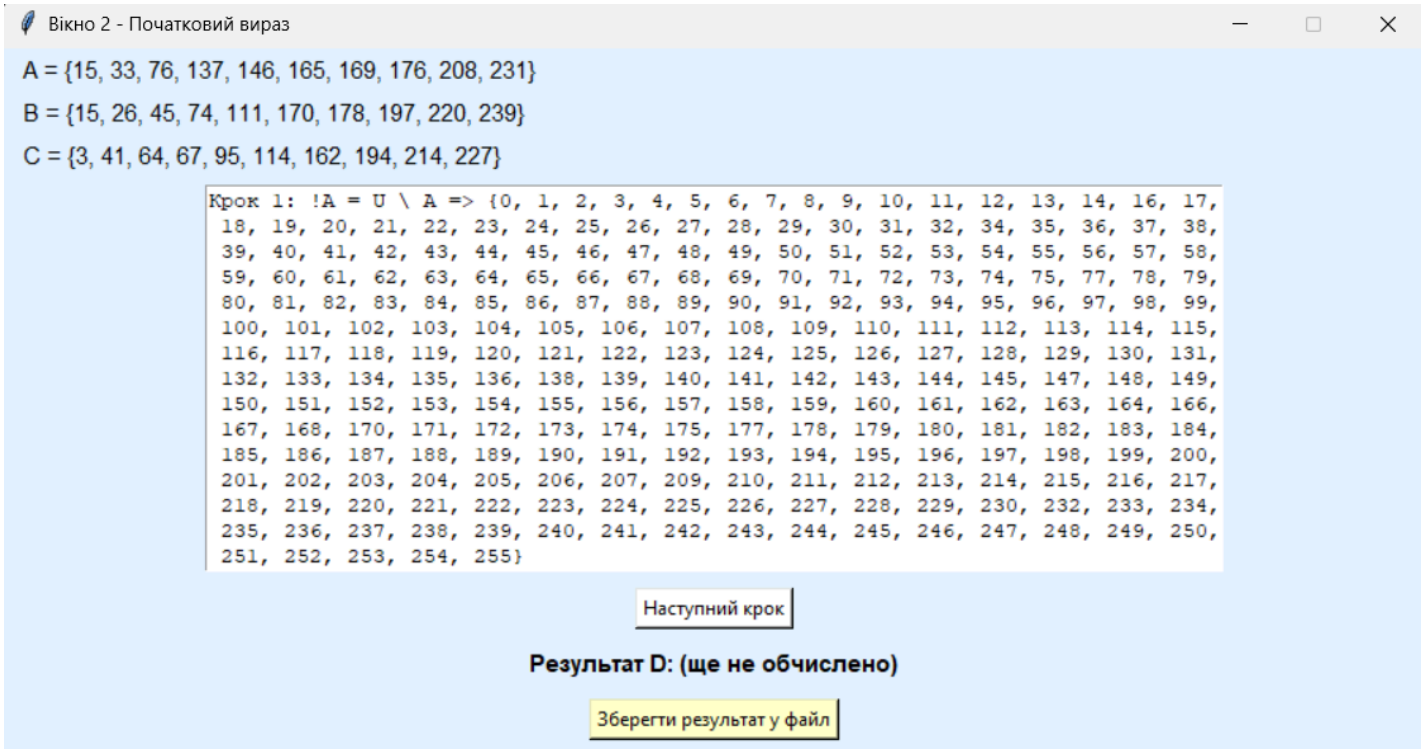


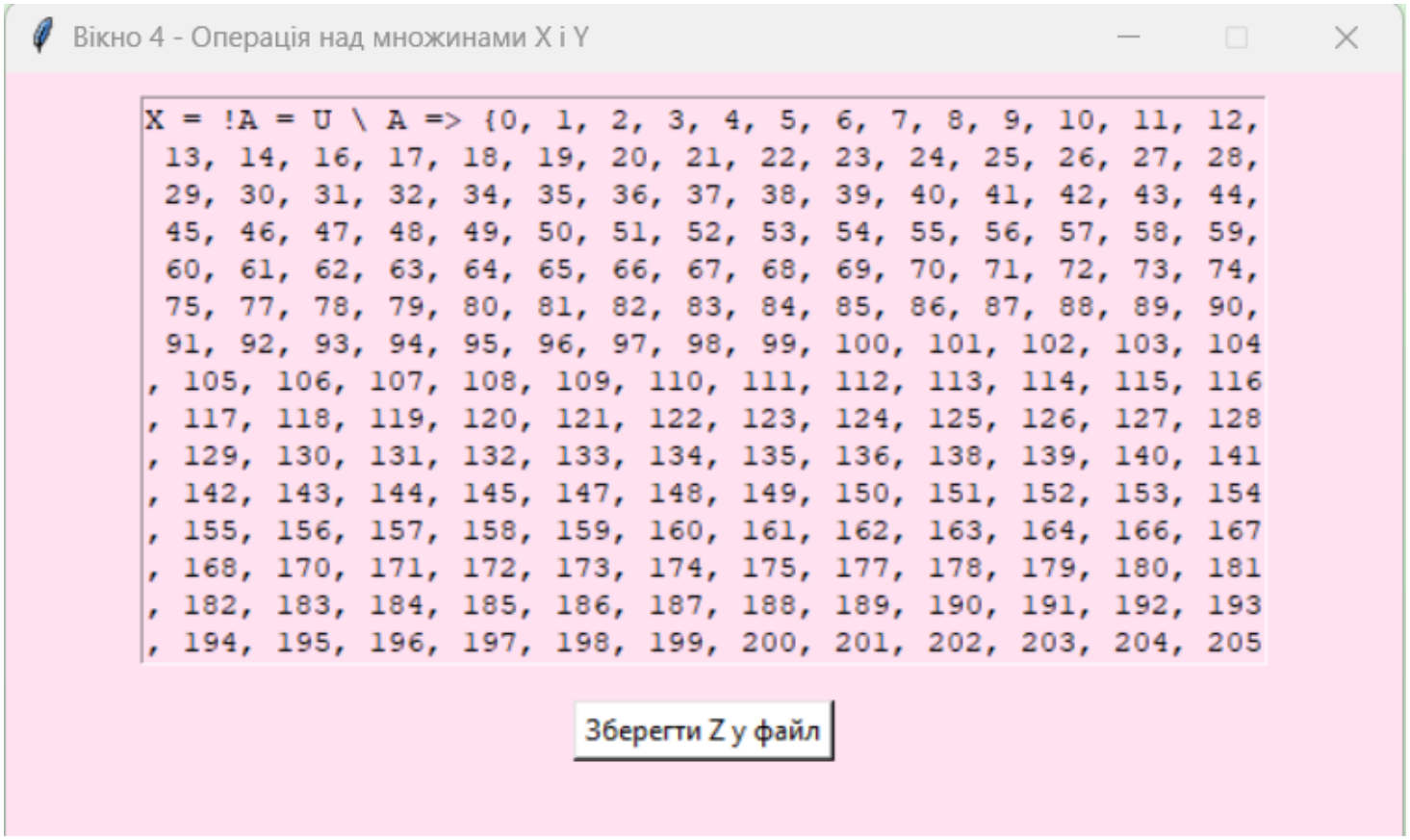
Вікно №2:



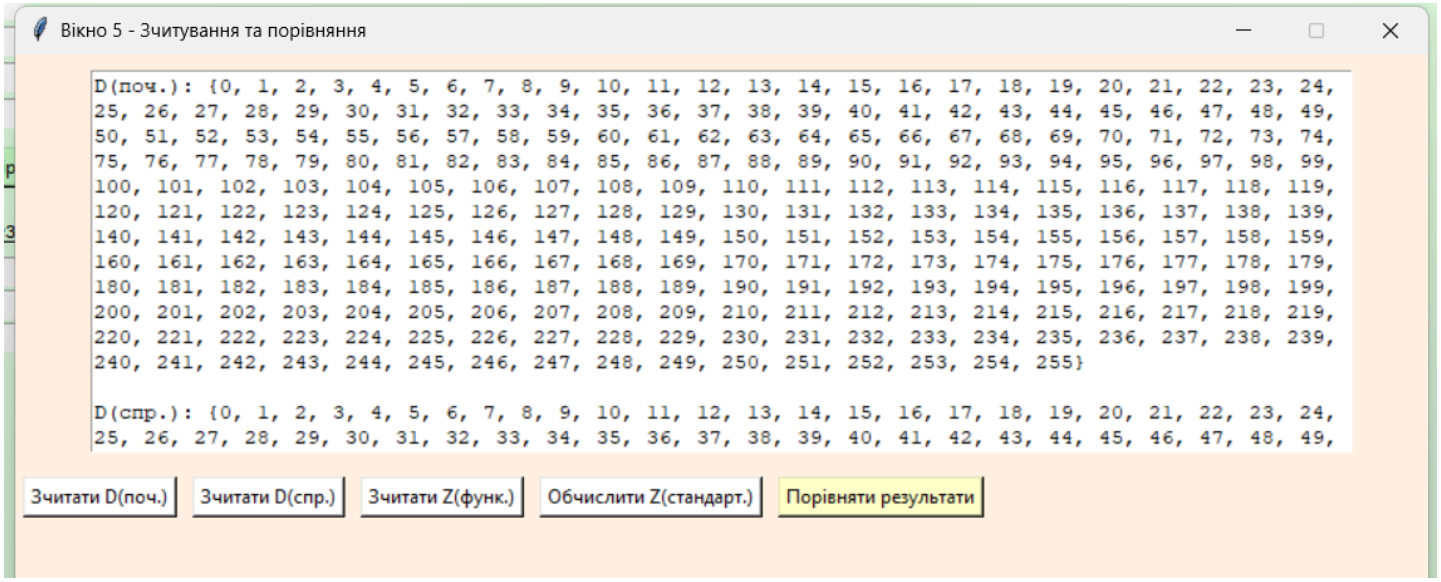

Вікно №3:

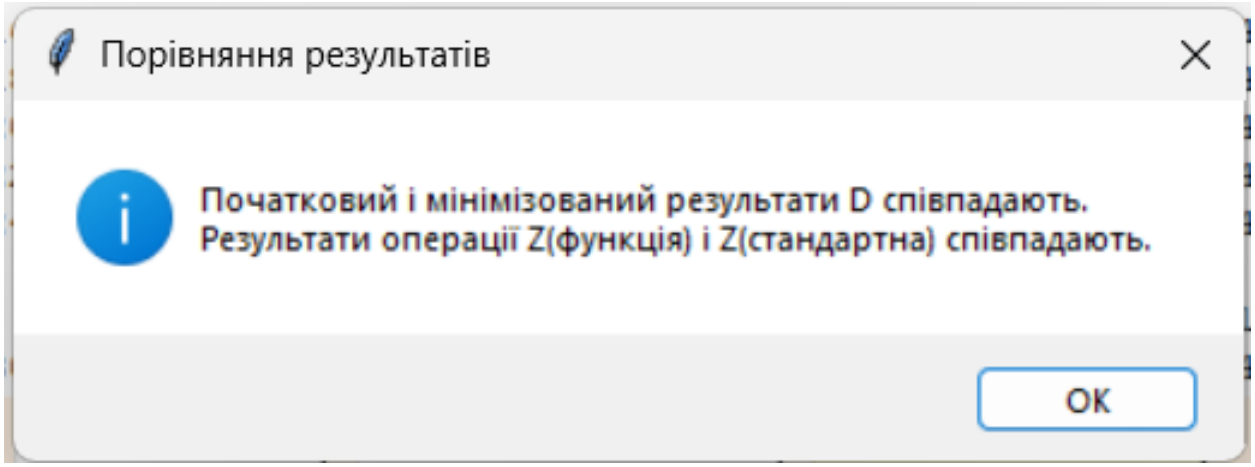


Вікно №4:




Вікно №5:



**Висновок:**

У ході виконання лабораторної роботи я створив програму на основі бібліотеки tkinter, яка виконує всі потрібні операції з множинами A, B, C та універсальною множиною U. У різних вікнах можна покроково обчислити початковий і спрощений вираз, автоматично отримати X та Y і побачити результат Z. Також передбачено зчитування з файлів, виконання стандартних операцій та порівняння отриманих результатів. Завдяки цьому я закріпив знання з роботи з множинами, логічними операціями та навички розробки графічного інтерфейсу в Python.