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»

Виконала:
студентка групи ІО-15
Кушнерик Є.О.
Залікова книжка №1508
Перевірив
Пономаренко А.М.

Київ 2021

Мета: ознайомитися з організацією графічного інтерфейсу на основі бібліотеки tkinter. Графічний інтерфейс (GUI) та його елементи. Модуль tkinter

Завдання

1.8. Загальний порядок виконання лабораторної роботи

1. Ознайомитися з теоретичним матеріалом. Опрацювати приклади.
2. Відповідно до свого варіанту визначити логічний вираз;
- написати програму, яка розв‘язує завдання - за допомогою стандартної бібліотеки *tkinter* запрограмувати відповідний графічний інтерфейс.
3. Скласти звіт і захистити його по роботі.

Захист роботи включає в себе демонстрацію працездатності програми на різних вхідних даних.

1.9. Вимоги до інтерфейсу

А) Програма повинна складатися з 3-х вікон.

Вміст вікна №1

1. Головне меню, яке повинно включати меню виклику вікна №2, вікна №3.
2. Віджети виводу П.І.Б студента, номера групи, номера у списку та віджет виводу результатів обчислення варіанту відповідно да програми, що задана у завданні – пункт загального порядку виконання лабораторної роботи).
3. Віджети для задавання кількості елементів множин A, B і C.
4. Віджети для формування випадковим чином множин A,B і C з заданою кількістю елементів.
5. Віджети, що дають можливість ручного вводу множин A,B і C.
6. Віджет для задавання діапазону цілих чисел, з яких будемо вибирати елементи множин.

Вміст вікна №2

1. Віджети для відображення елементів множин A, B і C.
2. Віджети запуску покрокового виконання виразу у відповідності з варіантом. Одним кроком вважати виконання однієї логічної операції.
3. Віджети відображення множин-операндів та множини-результату кожної логічної операції.
4. Віджет відображення множини D та віджет для виконання команди збереження даного результату у файлі.
5. Віджет, який містить зображення виразу 1 у відповідності з варіантом

Вміст вікна №3

1.Віджет для відображення елементів множин D.
2.Віджети для запуску випадкового генерування множини F з такими даними:
- кількість елементів множини F має дорівнювати кількості елементів множини D;
- мінімальний елемент множини F має дорівнювати мінімальному елементу множини D;
- максимальний елемент множини F має дорівнювати максимальному елементу множини D.
3.Віджет для відображення елементів множин F.
4.Віджет відображення множини X, та віджет для виконання команди збереження даного результату у файлі.
5. Віджет, який містить зображення виразу 2 у відповідності з варіантом

Варіант :

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

Код програми

```
from tkinter import *
from tkinter import messagebox
from random import randrange
root = Tk()
root.title("Головне меню")

class MainMenu:
    def __init__(self):
        root.update_idletasks()
        width = 1180
        height = 600
        frm_width = root.winfo_rootx() - root.winfo_x()
        win_width = width + 2 * frm_width
        titlebar_height = root.winfo_rooty() - root.winfo_y()
        win_height = height + titlebar_height + frm_width
        x = root.winfo_screenwidth() // 2 - win_width // 2
        y = root.winfo_screenheight() // 2 - win_height // 2
        root.geometry('{}x{}+{}+{}'.format(width, height, x, y))
        root.deiconify()
        mainmenu = Menu(root)
        root.config(menu=mainmenu)
        root.configure(background='lightblue')
        allmenu = Menu(mainmenu, tearoff=0)
        allmenu.add_command(label='Вікно 2', command=self.menu2)
        allmenu.add_command(label='Вікно 3', command=self.menu3)
        mainmenu.add_cascade(label='Вікно 1', menu=allmenu)
        mainmenu.add_command(label='Інфо',command=self.info)
        set_but = Button(root, bg='#00D1FF', width=20, height=2, text="Задати діапазон\n множин", font=("Helvetica",
10), command=self.MakeRange)
        set_but.grid(row=10, column=1, pady=20)

        univ_but = Button(root, bg='#00D1FF', width=20, height=2, text="Задати діапазон\n універсальної
множини",font=("Helvetica", 10), command=self.MakeRange2)
        univ_but.grid(row=10, column=2, pady=20)

        self.e_u_floor = Entry(root, width=4, font=("Helvetica", 20))
        self.e_u_floor.grid(row=10, column=1, sticky=E, pady=20)
        self.e_u_roof = Entry(root, width=4, font=("Helvetica", 20))
        self.e_u_roof.grid(row=10, column=2, sticky=W, pady=20)
```

```
        self.e_u_floor1 = Entry(root, width=4, font=("Helvetica", 20))
        self.e_u_floor1.grid(row=10, column=2, sticky=E, pady=20)
        self.e_u_roof1 = Entry(root, width=4, font=("Helvetica", 20))
        self.e_u_roof1.grid(row=10, column=3, sticky=W, pady=20)

        self.s1 = Scale(root, bg='#00D1FF', length=350, from_=0, to=200, tickinterval=25, resolution=1,
font=("Helvetica", 10))
        self.s2 = Scale(root, bg='#00D1FF', length=350, from_=0, to=200, tickinterval=25, resolution=1,
font=("Helvetica", 10))
        self.s3 = Scale(root, bg='#00D1FF', length=350, from_=0, to=200, tickinterval=25, resolution=1,
font=("Helvetica", 10))
        self.s1.grid(row=4, column=1, sticky=W, padx= 20)
        self.s2.grid(row=4, column=2, sticky=W)
        self.s3.grid(row=4, column=3, sticky=W, padx= 20)

        b_power_A = Button(root, width=15, bg='#00D1FF', text="Згенерувати A", font=("Helvetica", 10),
command=self.power_A)
        b_power_A.grid(row=6, column=1)
        b_power_B = Button(root, width=15, bg='#00D1FF', text="Згенерувати B", font=("Helvetica", 10),
command=self.power_B)
        b_power_B.grid(row=6, column=2)
        b_power_C = Button(root, width=15, bg='#00D1FF', text="Згенерувати C", font=("Helvetica", 10),
command=self.power_C)
        b_power_C.grid(row=6, column=3)

        self.e_A = Entry(root, width=40, font=("Helvetica", 12))
        self.e_A.grid(row=8, column=1, pady=20, padx=20)
        self.e_B = Entry(root, width=40, font=("Helvetica", 12))
        self.e_B.grid(row=8, column=2, pady=20)
        self.e_C = Entry(root, width=40, font=("Helvetica", 12))
        self.e_C.grid(row=8, column=3, pady=20, padx=20)

        b_hand_A = Button(root, width=15, bg='#00D1FF', text="Задати A", font=("Helvetica", 10),
command=self.hand_A)
        b_hand_A.grid(row=3, column=1, padx=20, pady=20)
        b_hand_B = Button(root, width=15, bg='#00D1FF', text="Задати B", font=("Helvetica", 10),
command=self.hand_B)
        b_hand_B.grid(row=3, column=2, padx=20, pady=20)
        b_hand_C = Button(root, width=15, bg='#00D1FF', text="Задати C", font=("Helvetica", 10),
command=self.hand_C)
```

```
    b_hand_C.grid(row=3, column=3, padx=20, pady=20)

def menu2(self):
    self.win2 = Toplevel(root)
    self.win2.configure(background = 'lightblue')
    width2 = 900
    height2 = 580
    frm_width2 = root.winfo_rootx() - root.winfo_x()
    win_width2 = width2 + 2 * frm_width2
    titlebar_height = root.winfo_rooty() - root.winfo_y()
    win_height = height2 + titlebar_height + frm_width2
    x2 = root.winfo_screenwidth() // 2 - win_width2 // 2
    y2 = root.winfo_screenheight() // 2 - win_height // 2
    self.win2.geometry('{}x{}+{}+{}'.format(width2, height2, x2, y2))
    self.win2.deiconify()
    self.win2.title("Меню 2")
    l_A = Label(self.win2, text="A = ", font=("Helvetica", 12), bg = 'lightblue')
    l_B = Label(self.win2, text="B = ", font=("Helvetica", 12), bg = 'lightblue')
    l_C = Label(self.win2, text="C = ", font=("Helvetica", 12), bg = 'lightblue')
    l_D = Label(self.win2, text="D = ", font=("Helvetica", 12), bg = 'lightblue')
    l_func = Label(self.win2, text="D = ((A∩!B)∪(B∩!A))∩(C∪B)∩C", font=("Helvetica", 15), bg='lightblue')
    l_step = Label(self.win2, text="Операція", font=("Helvetica", 12), bg = 'lightblue')
    l_A.grid(row=0, pady=20)
    l_B.grid(row=1)
    l_C.grid(row=2, pady=20)
    l_func.grid(row=3, columnspan=2, pady=5)
    l_D.grid(row=5, pady=20)
    l_step.grid(row=6, pady=20)

    self.tA = Text(self.win2, width=110, height=4, font=("Helvetica", 10))
    self.tB = Text(self.win2, width=110, height=4, font=("Helvetica", 10))
    self.tC = Text(self.win2, width=110, height=4, font=("Helvetica", 10))
    self.tD = Text(self.win2, width=110, height=4, font=("Helvetica", 10))
    self.t_step = Text(self.win2, width=110, height=4, font=("Helvetica", 10))
    self.tA.grid(row=0, column=1, pady=20)
    self.tB.grid(row=1, column=1)
    self.tC.grid(row=2, column=1, pady=20)
    self.tD.grid(row=5, column=1, pady=20)
    self.t_step.grid(row=6, column=1, padx=20)

    self.tA.insert(INSERT, list(self.A))
```

```
        self.tB.insert(INSERT, list(self.B))
        self.tC.insert(INSERT, list(self.C))
        b_step = Button(self.win2, bg='#00D1FF', width=15, text="Крок", font=("Helvetica",
10),command=self.func_long)
        b_step.grid(row=7, column=1, padx=20, sticky = W)
        b_save = Button(self.win2, bg='#00D1FF', width=15, text="Зберегти", font=("Helvetica", 10),
command=self.save_D)
        b_save.grid(row=7, column=1, pady=20, padx=20, sticky = E)

    def menu3(self):
        self.win3 = Toplevel(root)
        self.win3.configure(background='lightblue')
        self.win3.title("Меню 3")
        width = 1000
        height = 335
        frm_width = root.winfo_rootx() - root.winfo_x()
        win_width = width + 2 * frm_width
        titlebar_height = root.winfo_rooty() - root.winfo_y()
        win_height = height + titlebar_height + frm_width
        x = root.winfo_screenwidth() // 2 - win_width // 2
        y = root.winfo_screenheight() // 2 - win_height // 2
        self.win3.geometry('{}x{}+{}+{}'.format(width, height, x, y))
        self.win3.deiconify()
        self.l_D = Label(self.win3, text="D = ", font=("Arial", 13), padx = 10, bg = 'lightblue')
        self.l_F = Label(self.win3, text="F = ", font=("Arial", 13), bg = 'lightblue')
        self.l_fun2 = Label(self.win3, text="X = !FU!D", font=("Helvetica", 15), bg = 'lightblue')
        self.l_X = Label(self.win3, text="X = ", font=("Arial", 13), bg = 'lightblue')
        self.l_D.grid(row=0)
        self.l_F.grid(row=1)
        self.l_fun2.grid(row=2,column=1, pady = 20)
        self.l_X.grid(row=3)
        self.tD = Text(self.win3, width=110, height=4, font=("Helvetica", 10))
        self.tF = Text(self.win3, width=110, height=4, font=("Helvetica", 10))
        self.tX = Text(self.win3, width=110, height=4, font=("Helvetica", 10))
        self.tD.grid(row=0, column=1, pady = 20)
        self.tF.grid(row=1, column=1)
        self.tX.grid(row=3, column=1)

        b_hand_F = Button(self.win3, bg='#00D1FF', width=15, text="Задати F", font=("Helvetica", 10),
command=self.hand_F)
        b_hand_F.grid(row=1, column=2, padx = 20)
```

```
        b_readD = Button(self.win3, bg='#00D1FF', width=15, text="Прочитати", font=("Helvetica", 10),
command=self.readD)
        b_readD.grid(row=0, column=2)
        b_readX = Button(self.win3, bg='#00D1FF', width=15, text="Зберегти результати", font=("Helvetica", 10),
command=self.FoundX)
        b_readX.grid(row=3, column=2)
        self.tF.insert(INSERT, list(self.F))

    def info(self):
        self.a = Toplevel(root)
        self.a.config(bd=20, bg='lightblue')
        self.a.title("Інформація")
        self.a.resizable(False, False)
        Label(self.a, text="Группа ІО-15", font=("Helvetica", 15), bg='lightblue').pack()
        Label(self.a, text="Виконала - Кушнерик Є.О.", font=("Helvetica", 15), bg='lightblue').pack()
        Label(self.a, text="Варіант №8", font=("Helvetica", 15), bg='lightblue').pack()

    def readD(self):
        with open(r'D.txt', "r", encoding="utf-8") as r:
            self.tD.delete(1.0, END)
            self.tD.insert(INSERT, r.readline())

    def hand_A(self):
        self.A = str(self.e_A.get()).split()
        for i in range(len(self.A)):
            self.A[i] = int(self.A[i])
        try:
            if self.win2.state() == "normal" or self.win2.state() == "iconic" \
                        or self.win2.state() == "icon" or self.win2.state() == "withdrawn":
                self.tA.delete(1.0, END)
                self.tA.insert(INSERT, list(self.A))
        except AttributeError:
            pass
        except TclError:
            pass
        self.i = 0
        self.A = set(self.A)

    def hand_B(self):
        self.B = str(self.e_B.get()).split()
        for i in range(len(self.B)):
```

```
            self.B[i] = int(self.B[i])
        try:
            if self.win2.state() == "normal" or self.win2.state() == "iconic" \
                    or self.win2.state() == "icon" or self.win2.state() == "withdrawn":
                self.tB.delete(1.0, END)
                self.tB.insert(INSERT, list(self.B))
        except AttributeError:
            pass
        except TclError:
            pass
        self.i = 0
        self.B = set(self.B)

    def hand_C(self):
        self.C = str(self.e_C.get()).split()
        for i in range(len(self.C)):
            self.C[i] = int(self.C[i])
        try:
            if self.win2.state() == "normal" or self.win2.state() == "iconic" \
                    or self.win2.state() == "icon" or self.win2.state() == "withdrawn":
                self.tC.delete(1.0, END)
                self.tC.insert(INSERT, list(self.C))
        except AttributeError:
            pass
        except TclError:
            pass
        self.i = 0
        self.C= set(self.C)

    def hand_F(self):
        self.F = set()
        self.d = str(self.tD.get(1.0, END)).split()
        for i in range(len(self.d)):
            self.d[i] = int(self.d[i])
        self.mi = min(self.d)
        self.mx = max(self.d)
        while len(self.F) != len(self.d):
            self.F.add(self.mi)
            self.F.add(self.mx)
            self.F.add(randrange(self.mi, self.mx))
        self.tF.delete(1.0, END)
```

```
        self.tF.insert(INSERT, list(self.F))
        self.j = 0
        self.F = set(self.F)
        self.d = set(self.d)

    def MakeRange(self):
        self.rangeLeft = int(self.e_u_floor.get())
        self.rangeRight = int(self.e_u_roof.get())

    def MakeRange2(self):
        self.rangeLeft2 = int(self.e_u_floor1.get())
        self.rangeRight2 = int(self.e_u_roof1.get())

    def power_A(self):
        A = set()
        if (self.e_u_floor.index("end") == 0 and self.e_u_roof.index("end") == 0):
            while len(A) != self.s1.get():
                A.add(randrange(256))
            self.e_A.delete(0, END)
            self.e_A.insert(INSERT, list(A))
        else:
            if (self.rangeLeft < self.rangeRight):
                while len(A) != self.s1.get():
                    A.add(randrange(self.rangeLeft, self.rangeRight+1))
                self.e_A.delete(0, END)
                self.e_A.insert(INSERT, list(A))
            else:
                messagebox.showerror("Error", "Ліва границя не може бути більшою за праву")

    def power_B(self):
        B = set()
        if (self.e_u_floor.index("end") == 0 and self.e_u_roof.index("end") == 0):
            while len(B) != self.s2.get():
                B.add(randrange(256))
            self.e_B.delete(0, END)
            self.e_B.insert(INSERT, list(B))
        else:
            if (self.rangeLeft < self.rangeRight):
                while len(B) != self.s2.get():
                    B.add(randrange(self.rangeLeft, self.rangeRight+1))
                self.e_B.delete(0, END)
```

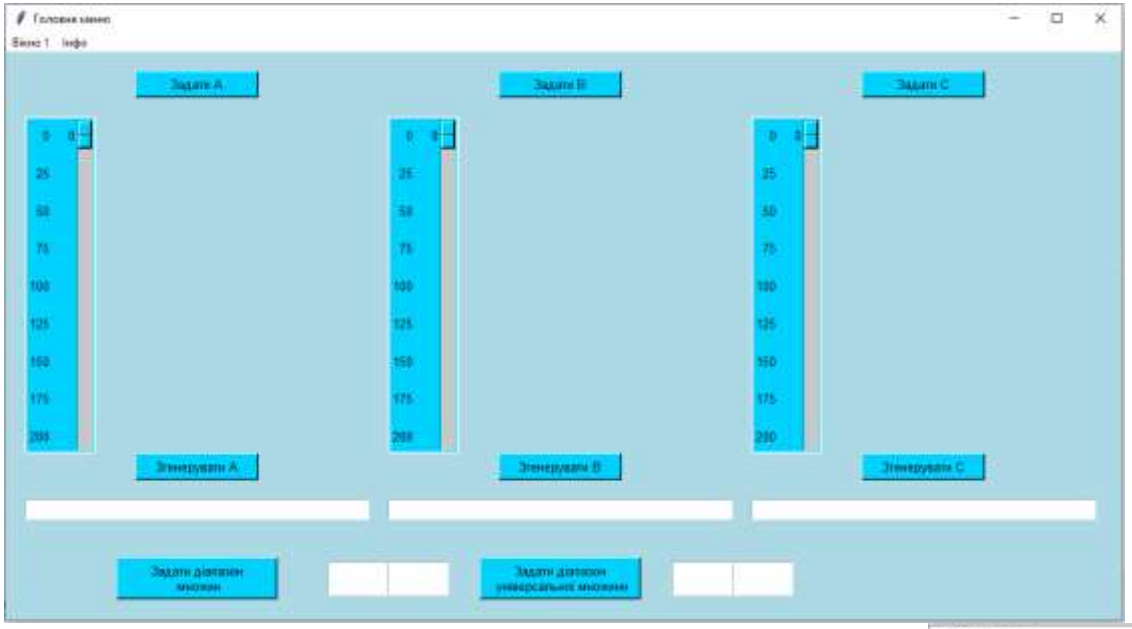
```
            self.e_B.insert(INSERT, list(B))
        else:
            messagebox.showerror("Error", "Ліва границя не може бути більшою за праву")

def power_C(self):
    C = set()
    if (self.e_u_floor.index("end") == 0 and self.e_u_roof.index("end") == 0):
        while len(C) != self.s3.get():
            C.add(randrange(256))
        self.e_C.delete(0, END)
        self.e_C.insert(INSERT, list(C))
    else:
        if (self.rangeLeft < self.rangeRight):
            while len(C) != self.s3.get():
                C.add(randrange(self.rangeLeft, self.rangeRight + 1))
            self.e_C.delete(0, END)
            self.e_C.insert(INSERT, list(C))
        else:
            messagebox.showerror("Error", "Ліва границя не може бути більшою за праву")

def FoundX(self):
    if self.j == 0:
        X = set(range(self.rangeLeft2, self.rangeRight2))
        X.difference_update(self.d)
        X.difference_update(self.F)
        self.new_X = "!FU!D=!{0}U!{1}= {2}\n".format(self.d , self.F , X)
        self.tX.delete(1.0, END)
        self.tX.insert(INSERT, self.new_X)
        self.j += 1
        self.save_x()
    else:
        self.tX.delete(1.0, END)
        self.j = 0

def func_long(self):
    if self.i == 0:
        self.D_long = "({0}) \n".format(set.union(self.B, self.C))
        self.tD.delete(1.0, END)
        self.tD.insert(INSERT, self.D_long)
        self.t_step.delete(1.0, END)
        self.t_step.insert(INSERT, "(CUB)")
```

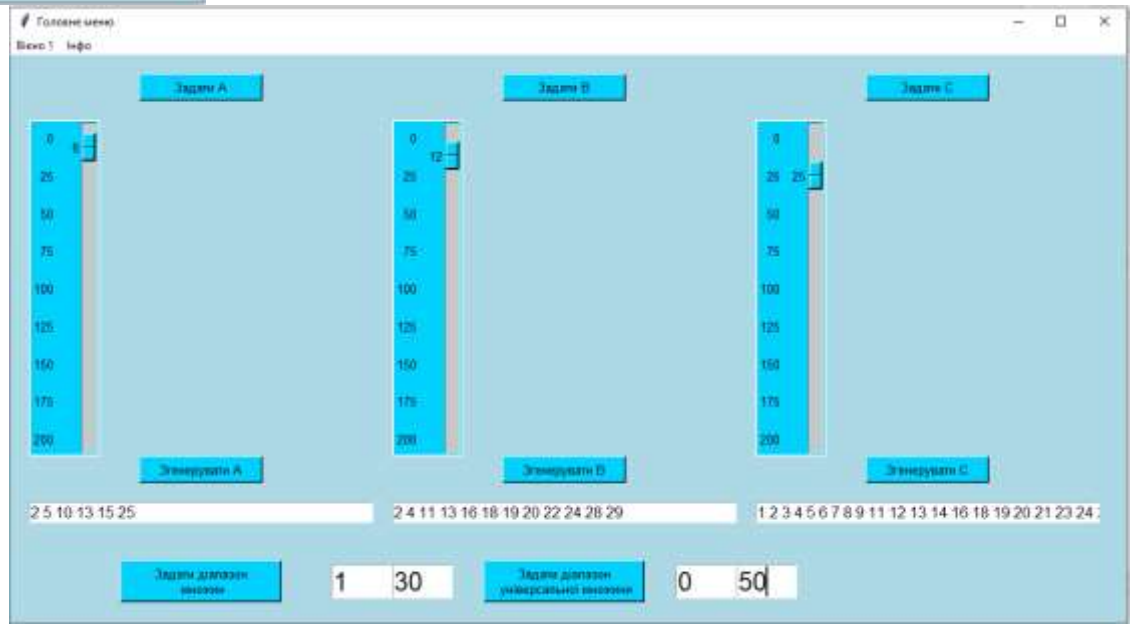
```
        self.set_one = set.union(self.B, self.C)
        self.i += 1

    elif self.i == 1:
        self.D_long = "({0}) \n".format(set.intersection(self.set_one, self.C))
        self.tD.delete(1.0, END)
        self.tD.insert(INSERT, self.D_long)
        self.t_step.delete(1.0, END)
        self.t_step.insert(INSERT, "(C∪B)∩C")
        self.set_two = set.intersection(self.set_one, self.C)
        self.i += 1
    elif self.i == 2:
        self.D_long = "({0}) \n".format(set.intersection(self.A, self.C))
        self.tD.delete(1.0, END)
        self.tD.insert(INSERT, self.D_long)
        self.t_step.delete(1.0, END)
        self.t_step.insert(INSERT, "(A∩!B)=(A∩C)")
        self.set_three = set.intersection(self.A, self.C)
        self.i += 1
    elif self.i == 3:
        self.D_long = "({0}) \n".format(set.intersection(self.B, self.C))
        self.tD.delete(1.0, END)
        self.tD.insert(INSERT, self.D_long)
        self.t_step.delete(1.0, END)
        self.t_step.insert(INSERT, "(B∩!A)=(B∩C)")
        self.set_four = set.intersection(self.B, self.C)
        self.i += 1
    elif self.i == 4:
        self.D_long = "({0}) \n".format(set.union(self.set_three, self.set_four))
        self.tD.delete(1.0, END)
        self.tD.insert(INSERT, self.D_long)
        self.t_step.delete(1.0, END)
        self.t_step.insert(INSERT, "(A∩!B)∪(B∩!A)=(A∩C)∪(B∩C)")
        self.set_five = set.union(self.set_three, self.set_four)
        self.i += 1
    elif self.i == 5:
        self.D_long = ""
        for i in (list(set.intersection(self.set_five, self.set_two))):
            self.D_long = self.D_long + str(i) + " "
        self.tD.delete(1.0, END)
        self.tD.insert(INSERT, self.D_long)
```

```
            self.t_step.delete(1.0, END)
            self.t_step.insert(INSERT, "((A∩!B)∪(B∩!A))∩(C∪B)∩C")
            self.i += 1
        else:
            self.tD.delete(1.0, END)
            self.t_step.delete(1.0, END)
            self.i = 0

    def save_D(self):
        if "{" not in self.D_long and "(" not in self.D_long:
            with open(r'D.txt', "a", encoding="utf-8") as w:
                w.write(str(self.D_long) + "\n")
    def save_x(self):
            with open(r'X.txt', "a", encoding="utf-8") as w:
                w.write(str(self.new_X) + "\n")

start = MainMenu()
root.mainloop()
```

Результат програми

Головне меню





Меню 2

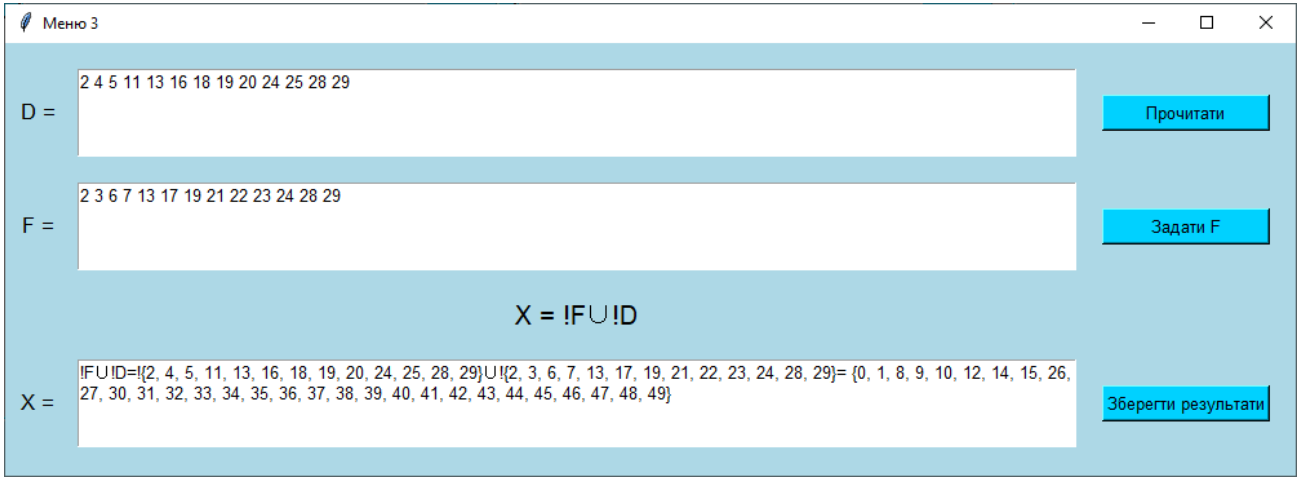

Меню 2
A =
2 5 10 13 15 25
B =
2 4 11 13 16 18 19 20 22 24 28 29
C =
1 2 3 4 5 6 7 8 9 11 12 13 14 16 18 19 20 21 23 24 25 27 28 29 30
D = ((A∩!B)∪(B∩!A))∩(C∪B)∩C
D =
Операція
Крок
Зберегти


Меню 2
A =
2 5 10 13 15 25
B =
2 4 11 13 16 18 19 20 22 24 28 29
C =
1 2 3 4 5 6 7 8 9 11 12 13 14 16 18 19 20 21 23 24 25 27 28 29 30
D = ((A∩!B)∪(B∩!A))∩(C∪B)∩C
D =
{(1, 2, 3, 4, 5, 6, 7, 8, 9, 11, 12, 13, 14, 16, 18, 19, 20, 21, 22, 23, 24, 25, 27, 28, 29, 30)}
Операція
(C∪B)
Крок
Зберегти

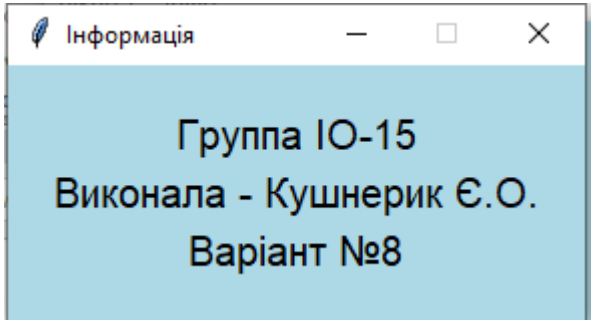

Меню 2
A =
2 5 10 13 15 25
B =
2 4 11 13 16 18 19 20 22 24 28 29
C =
1 2 3 4 5 6 7 8 9 11 12 13 14 16 18 19 20 21 23 24 25 27 28 29 30
D = ((A∩!B)∪(B∩!A))∩(C∪B)∩C
D =
{(2, 4, 5, 11, 13, 16, 18, 19, 20, 24, 25, 28, 29)}
Операція
(A∩!B)∪(B∩!A)=(A∩C)∪(B∩C)
Крок
Зберегти


Меню 2
A =
2 5 10 13 15 25
B =
2 4 11 13 16 18 19 20 22 24 28 29
C =
1 2 3 4 5 6 7 8 9 11 12 13 14 16 18 19 20 21 23 24 25 27 28 29 30
D = ((A∩!B)∪(B∩!A))∩(C∪B)∩C
D =
2 4 5 11 13 16 18 19 20 24 25 28 29
Операція
((A∩!B)∪(B∩!A))∩(C∪B)∩C
Крок
Зберегти

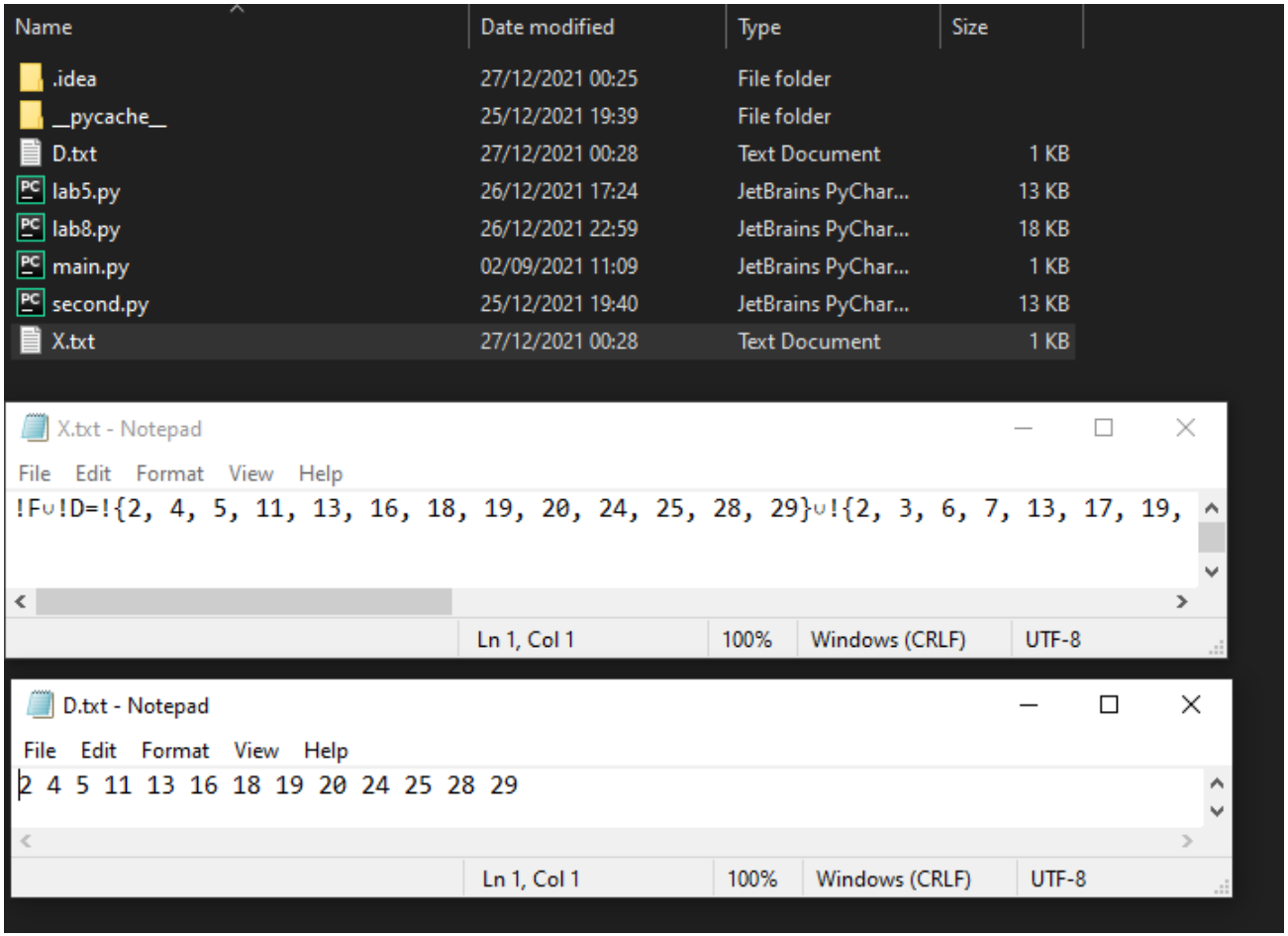
Меню 3



Вікно Інфо



Вміст створеного файлу с результатами операцій D та X



Висновок :

Я ознайомилася з організацією графічного інтерфейсу на основі бібліотеки tkinter. Розглянула графічний інтерфейс (GUI) та його елементи. Модуль tkinter.Розробили програму на основі наданого завдання в графічній оболонці.