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 ІВ-71
Мазан Ян Владиславович
Залікова книжка
№7109
Перевірив:
Саверченко Василь Григорович

Київ-2018

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

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

Загальне завдання:

1. Повторити матеріал: «Бібліотека `tkinter` (віджети)» та виконати лабораторну роботу з застосуванням графічного інтерфейсу.
2. Спростити логічний вираз з застосуванням тотожностей алгебри множин.
3. В окремому модулі написати функцію обчислення початкового логічного виразу (1), вибраного відповідно до індивідуального варіанта.
4. В окремому модулі написати функцію обчислення спрощеного логічного виразу.
5. В окремому модулі написати функцію виконання логічної операції (2), вибраної відповідно до індивідуального варіанта.
6. В окремому модулі виконати порівняння результатів:
   А) обчислення початкового та спрощеного виразу
   Б) виконання логічної операції Вашою функцією та відповідною стандартною логічною операцією або функцією `Python`.

Варіант виразу відповідно до індивідуального завдання:

i = 9
G = 71
Z = (9 + 71mod60)mod30 +1 = 21

21	(1)	$D = A \cap \left(\overline{B} \cup C\right) \cup \left(\overline{A} \cap C\right)$
	(2)	$X = C; Y = B; Z = X \setminus Y$

Теоретичні відомості:

Множина – є сукупність визначених об'єктів, різних між собою, об'єднаних за певною ознакою чи властивістю.

Множини позначають **великими** латинськими буквами. Об'єкти, що складають множини, називають елементами і позначають **малими** буквами латинського алфавіту.

На практиці часто застосовують такі позначення:

N – множина натуральних чисел;
P – множина додатних цілих чисел;
Z – множина додатних і від'ємних цілих чисел, включаючи нуль;
Q – множина раціональних чисел;
R – множина дійсних чисел.

Якщо множина не містить жодного елемента, її називають порожньою і позначають $\varnothing$.

Підмножина. Множину A називають підмножиною (або *включенням*) множини B $(A \subseteq B)$, якщо кожен елемент множини A є елементом множини B, тобто, якщо $x \in A$, то $x \in B$. Якщо $A \subseteq B$ й $A \neq B$, то A називають строгою підмножиною й позначають $A \subset B$.

Рівність множин. Дві множини рівні $(A = B)$, якщо всі їхні елементи збігаються. Множини A і B рівні, якщо $A \subseteq B$ і $B \subseteq A$.

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

1.2. Операції над множинами

Об'єднання. *Об'єднанням* множин A і B називають множину, що складається із всіх тих елементів, які належать хоча б одній з множин A або B. Об'єднання множин A і B позначають $A \cup B$. Це визначення рівносильне наступному: $A \cup B = \{x | x \in A \text{ або } x \in B\}$.

Перетин. *Перетином* множин A і B називають множину, що складається із всіх тих елементів, які належать як множині A, так і множині B. Перетин множин A і B позначають $A \cap B$. Це визначення рівносильне наступному: $A \cap B = \{x | x \in A \ i \ x \in B\}$.

Доповнення. *Доповненням (*або *абсолютним доповненням)* множини A називають множину, що складається із всіх елементів універсальної множини, які не належать A. Доповнення множини A позначають $\overline{A}$. Це визначення рівносильне наступному: $\overline{A} = U - A = \{x | x \in U \ и \ x \notin A\}$.

Різниця. *Різницею* множин A й B *(*або *відносним доповненням)* називають множину, що складається із всіх елементів множини A, які не належать B. Різницю множин A і B позначають $A - B$ або $A \setminus B$. Це визначення рівносильне наступному: $A - B = \{x | x \in A \ и \ x \notin B\}$.

Способи задавання множин:

- ***перерахуванням,*** тобто списком всіх елементів. Такий спосіб задавання прийнятний тільки при задаванні скінченних множин. Позначення списку – у фігурних дужках. Наприклад, множина, що з перших п'яти простих чисел $A = \{2,3,5,7,11\}$. Множина спортсменів університетської хокейної команди: $B = \{Іванов, Петров, Сидоров, Бубликов, Сироєжкін, Волосюк\}$;

- ***процедурою***, що породжує і описує спосіб одержання елементів множини із уже отриманих елементів або з інших об'єктів. Наприклад, множина усіх цілих чисел, що є степенями двійки $M_{2^n}, n \in N$, де N - множина натуральних чисел, може бути представлена породжуючою процедурою, заданою двома правилами, названими рекурсивними: а) $1 \in M_{2^n}$; б) якщо $m \in M_{2^n}$, тоді $2m \in M_{2^n}$;

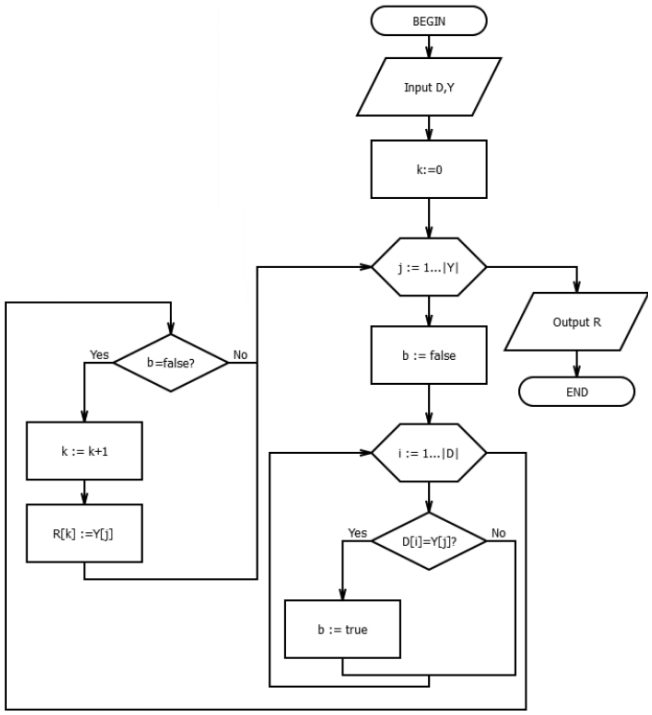
<u>Докладний опис спрощення логічного виразу з посиланням на тотожності алгебри множин, що застосовувалися при спрощенні:</u>

$$D = A \cap \left(\overline{B} \cup C\right) \cup \left(\overline{A} \cap C\right) = \left|\text{застосування закону дистрибутивності}\right| =$$

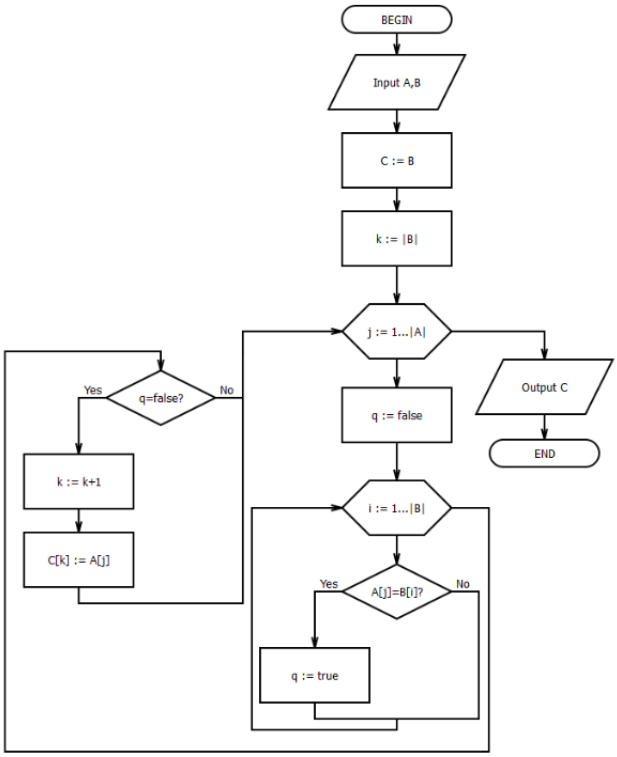
$$= \left(A \cap \overline{B}\right) \cup (A \cap C) \cup \left(\overline{A} \cap C\right) = \left|\text{застосування закону склеювання}\right| =$$

$$= \left(A \cap \overline{B}\right) \cup C = \left|\text{застосування закону комутативності}\right| = \left(\overline{B} \cap A\right) \cup C$$

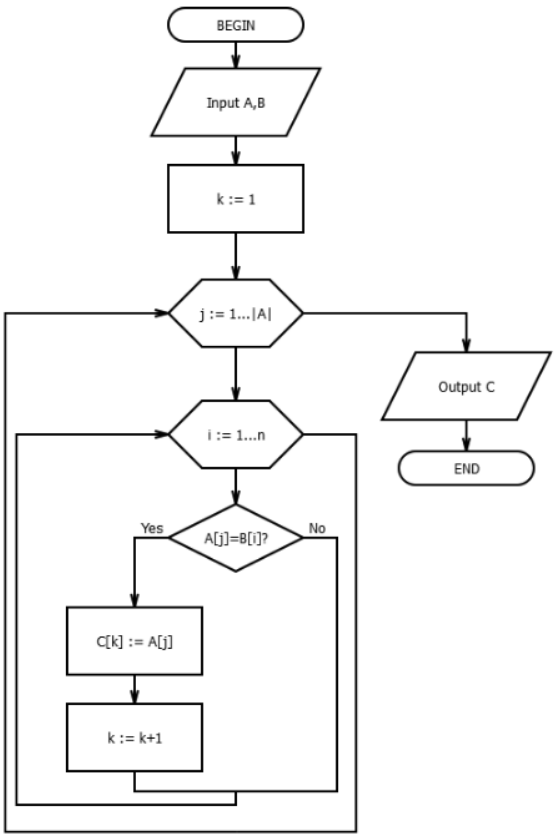
6. Блок-схеми, які відповідають алгоритмам, що використані в лабораторній роботі.



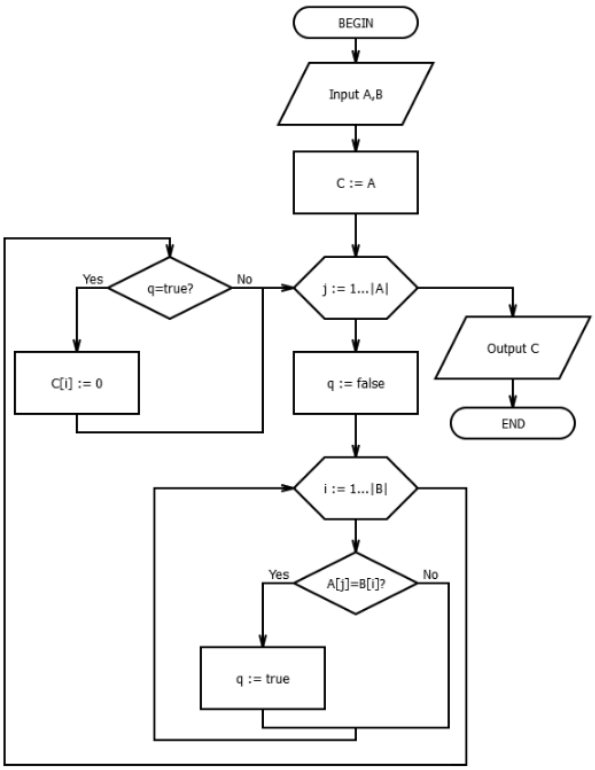
Блок-схема операції доповнення множини до універсальної



Блок-схема операції об'єднання двох множин



. Блок-схема операції перетину двох множин



Блок-схема операції різниці множин

7. Роздруківка того фрагменту тексту програми, який написаний індивідуально.
Файл SetsCreation.py:

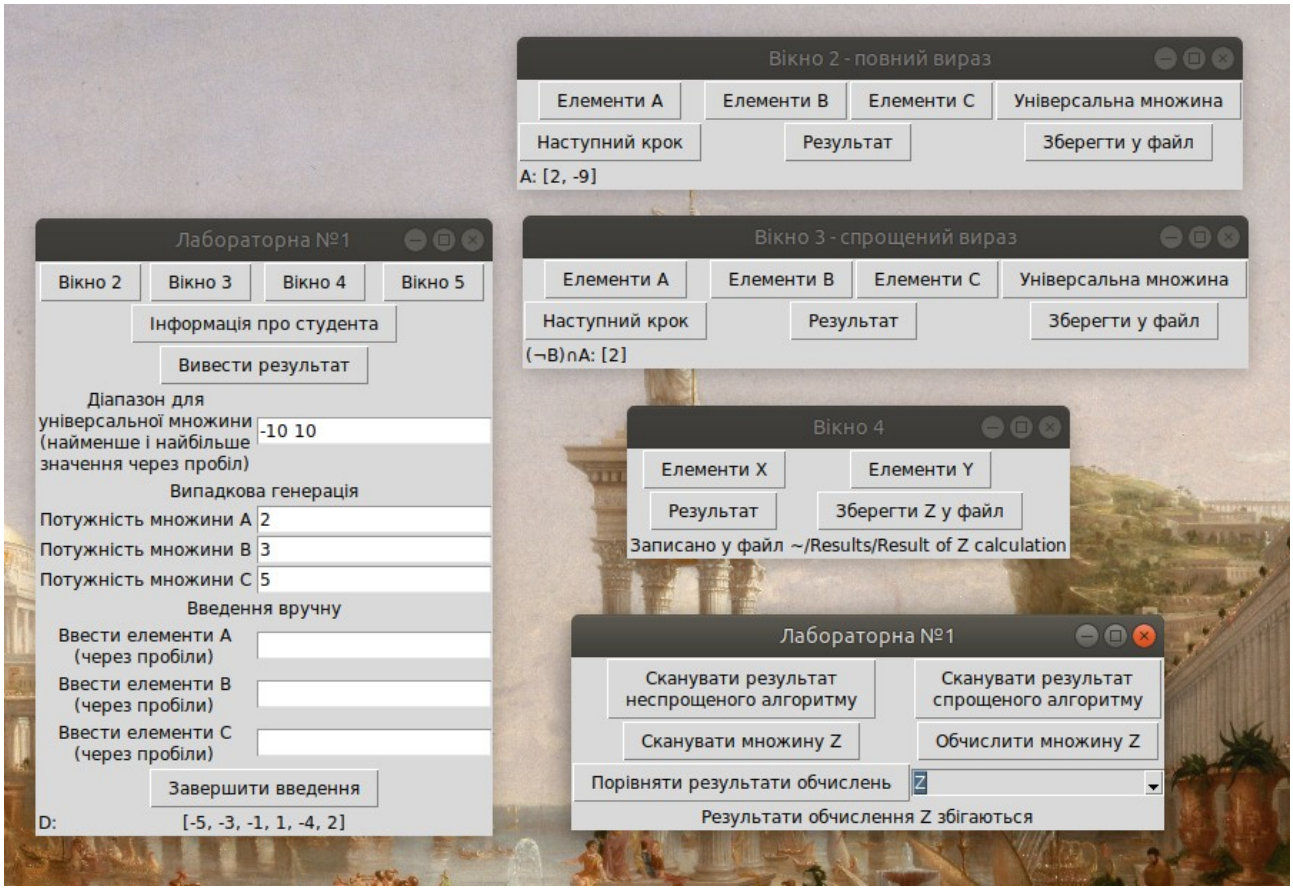
```
class Set():
    def __init__(self,values):
        self.values = values

    def __call__(self):
        return self.values

    def __str__(self):
        return str(self.values)

    def copy(self):
        return Set([i for i in self.values])

    def inversion(self,Universal: "Set"):
        k=0
        Result = []
        for j in range(len(Universal.values)):
            b = False
            for i in range(len(self.values)):
                if self.values[i] == Universal.values[j]: b = True
            if b == False:
                k+=1
                Result.append(Universal.values[j])
        return Set(Result)

    def union(self,B: "Set"):
        Result = B.values
        k = len(B.values)
        for j in range(len(self.values)):
            q = False
            for i in range(len(B.values)):
                if self.values[j] == B.values[i]: q = True
            if q == False:
                k+=1
                Result.append(self.values[j])
        return Set(Result)

    def intersection(self,B: "Set"):
        k=0
        Result = []
        for j in range(len(self.values)):
            for i in range(len(B.values)):
                if self.values[j] == B.values[i]:
                    Result.append(self.values[j])
                    k+=1
        return Set(Result)

    def difference(self,B: "Set"):
        Result = self.values
        for j in range(len(self.values)):
            q = False
            for i in range(len(B.values)):
                if self.values[j] == B.values[i]: q = True
                else: continue
            if q == True:
                Result[j] = None
        #видалення об'єктів NoneType
        Result = [i for i in Result if not i == None]
```

```
    return Set(Result)
```

Файл Calculations.py:

```
def calculate_simplified(set_A, set_B, set_C, Universal):
    return set_B.copy().inversion(Universal.copy()).intersection(set_A.copy()).union(set_C.copy())

def calculate_Z(set_B,set_C):
    set_X = set_C.copy()
    set_Y = set_B.copy()
    return set_X.difference(set_Y)

def calculate_full(set_A, set_B, set_C, Universal):
    return
set_A.copy().intersection(set_B.copy().inversion(Universal.copy()).union(set_C.copy())).union(set_A.copy().inversion(
Universal.copy()).intersection(set_C.copy()))


#Покрокове виконання спрощеного

def operation_short1(set_A, set_B, set_C, Universal):
    return ["¬B",set_B.inversion(Universal)]

def operation_short2(set_A, set_B, set_C, Universal):
    return ["(¬B)∩A", operation_short1(set_A, set_B, set_C, Universal)[1].intersection(set_A)]

def operation_short3(set_A, set_B, set_C, Universal):
    return ["D = [(¬B)∩A]∪C",operation_short2(set_A, set_B, set_C, Universal)[1].union(set_C)]


#Покрокове виконання довгого

def operation_long1(set_A, set_B,set_C, Universal):
    return ["¬B",set_B.copy().inversion(Universal.copy())]

def operation_long2(set_A, set_B,set_C, Universal):
    return ["¬A", set_A.copy().inversion(Universal.copy())]

def operation_long3(set_A, set_B,set_C, Universal):
    return ["¬B∪C", operation_long1(set_A.copy(), set_B.copy(),set_C.copy(), Universal.copy())
[1].union(set_C.copy())]

def operation_long4(set_A, set_B,set_C, Universal):
    return ["(¬A)∩C", operation_long2(set_A.copy(), set_B.copy(),set_C.copy(), Universal.copy())
[1].intersection(set_C.copy())]

def operation_long5(set_A, set_B, set_C, Universal):
    return ["A∩(¬B∪C)", set_A.copy().intersection(operation_long3(set_A.copy(), set_B.copy(),set_C.copy(),
Universal.copy())[1])]

def operation_long6(set_A, set_B, set_C, Universal):
    return ["D = [A∩(¬B∪C)]∪[(¬A)∩C]", operation_long5(set_A.copy(), set_B.copy(), set_C.copy(),
Universal.copy())[1].union(operation_long4(set_A.copy(), set_B.copy(),set_C.copy(), Universal.copy())[1])]
```

8. Роздруківка результатів виконання програми з контрольним прикладом



9. Аналіз результатів та висновки.

Під час виконання даної лабораторної роботи я навчився навичкам роботи з Tkinter у Python, вивчив способи задання множин та операції над ними. Під час виконання роботи виникали деякі проблеми із перетворенням програми в об’єктний код із процедурного для спрощення її розуміння.