|  |  |
| --- | --- |
|  | **Министерство науки и высшего образования Российской Федерации**  Калужский филиал  федерального государственного бюджетного  образовательного учреждения высшего образования  ***«Московский государственный технический университет***  ***имени Н.Э. Баумана***  ***(национальный исследовательский университет)»***  ***(КФ МГТУ им. Н.Э. Баумана)*** |

|  |  |
| --- | --- |
| **ФАКУЛЬТЕТ** | ***ИУК «Информатика и управление»*** |
| **КАФЕДРА** | ***ИУК4 «Программное обеспечение ЭВМ, информационные технологии»*** |



**ЛАБОРАТОРНАЯ РАБОТА №2**

**«Замыкание отношений»**

|  |
| --- |
| **ДИСЦИПЛИНА: «Дискретная математика»** |

|  |  |
| --- | --- |
| Выполнил: студент гр.ИУК4-32Б | \_\_\_\_\_\_\_\_\_ (Зудин Д. В.)  (Подпись) (Ф.И.О.) |
| Проверил: | \_\_\_\_\_\_\_\_\_ (Никитенко У. В.)  (Подпись) (Ф.И.О.) |

|  |  |  |
| --- | --- | --- |
| Дата сдачи (защиты): |  | |
| Результаты сдачи (защиты):  - Бальная оценка:    - Оценка: | |

Калуга, 2022 г.

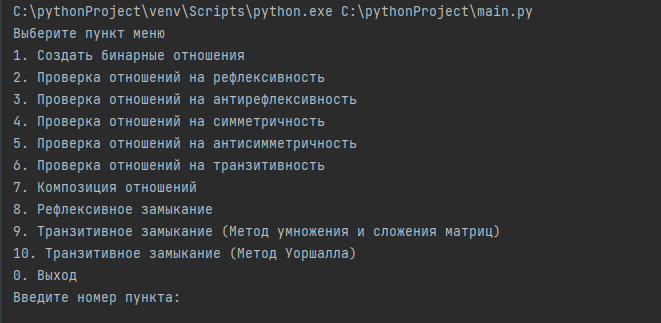
Цели исследование свойств бинарных отношений, построение замыкания отношений

Задачи. определить какими свойствами обладают бинарные отношения, построение композиции отношений, построение рефлексивного, симметричного и транзитивного замыкания отношений

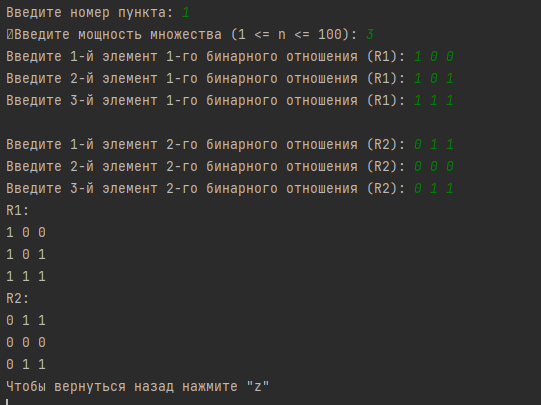
ЗАДАНИЕ

Написать программу, в которой: Дано число n и два бинарных отношения на множестве размера n. Для каждого из этих отношений определяется, являются ли они рефлексивными, антирефлексивными, симметричными, антисимметричными и транзитивными, а так же найдите их композицию. Найдите рефлексивное, симметричное транзитивное (двумя способами: с помощью умножения и сложения матриц; с помощью алгоритма Уоршалла) замыкание.

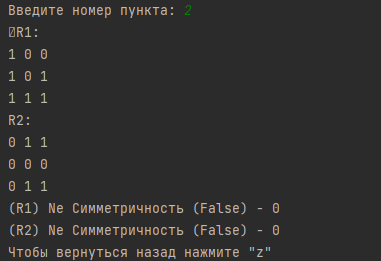
Этапы выполнения программы:

1.Вывод основного меню 

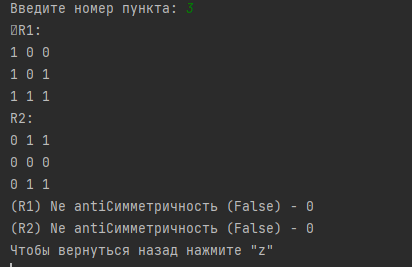
2.Создание бинарных отношений



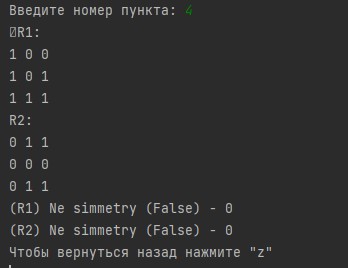
2. Проверка отношений на рефлексивность



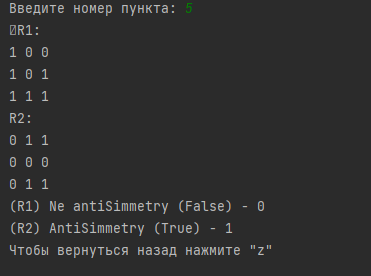
3. Проверка отношений на антирефлексивность



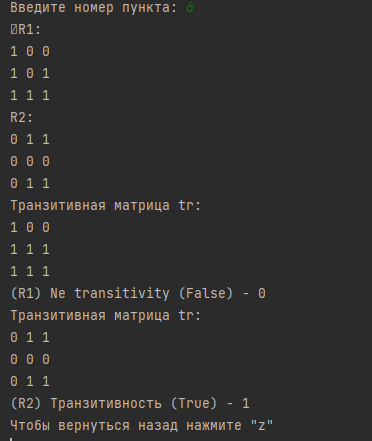
4. Проверка отношений на симметричность



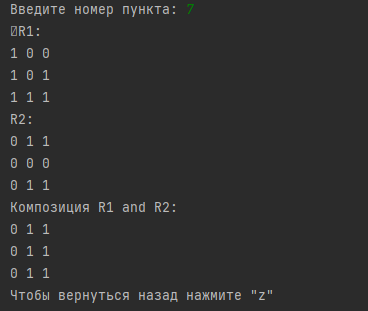
5. Проверка отношений на антисимметричность



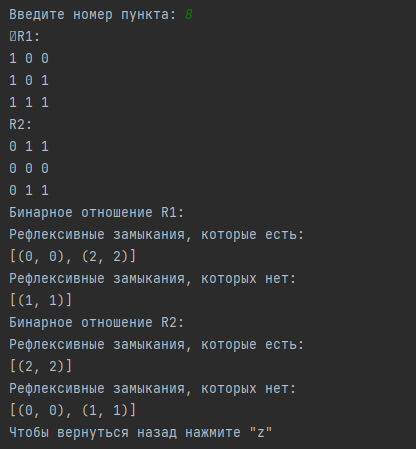
6. Проверка отношений на транзитивность



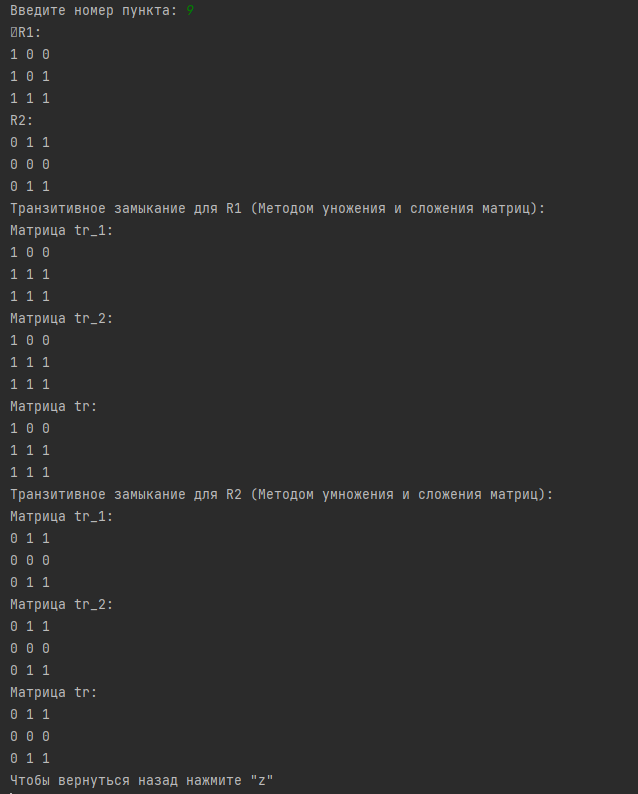
7. Композиция отношений



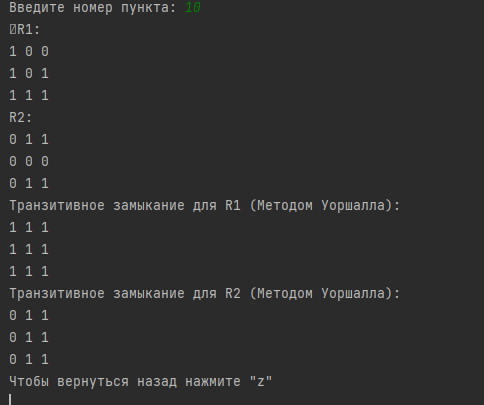
8. Рефлексивное замыкание



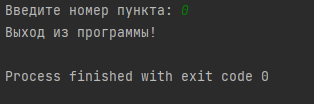
9. Транзитивное замыкание (Метод умножения и сложения матриц)



10. Транзитивное замыкание (Метод Уоршалла)



0. Выход



Листинг программы:

import os

def clear():

os.system("clear")

def ПечатьОтношений(m, n):

print("R1:")

for line in m:

print(\*line)

print("R2:")

for line in n:

print(\*line)

for i in range(len(m)):

if m[i][i] != 1:

return False

return True

def Симметричность(m):

for i in range(len(m)):

if m[i][i] != 1:

return False

return True

def АнтиСимметричность(m):

for i in range(len(m)):

if m[i][i] != 0:

return False

return True

def Симетр(m):

for i in range(len(m)):

for j in range(len(m)):

if i != j:

if m[i][j] != m[j][i]:

return False

return True

def АнтиСиметр(m):

n = len(m)

nm = [[0 for \_ in range(n)] for \_ in range(n)]

for i in range(n):

for j in range(n):

nm[i][j] = m[i][j] \* m[j][i]

flag = True

for i in range(len(m)):

for j in range(len(m)):

if i != j:

if nm[i][j] != 0:

flag = False

return flag

def Транзитивность(m):

length = len(m)

tr = [[0 for \_ in range(0, length)] for \_ in range(0, length)]

flag = True

for i in range(len(m)):

for j in range(len(m)):

for k in range(len(m)):

tr[i][j] = (tr[i][j] | (m[i][k] & m[k][j]))

print("Транзитивная матрица tr:")

for line in tr:

print(\*line)

if (m != tr):

flag = False

return flag

def Композиция(m, n):

length = len(m)

comp = [[0 for \_ in range(0, length)] for \_ in range(0, length)]

for i in range(len(m)):

for j in range(len(m)):

for k in range(len(m)):

comp[i][j] = comp[i][j] + m[i][k] \* n[k][j]

if comp[i][j] != 0:

comp[i][j] = 1

print("Композиция R1 and R2:")

for line in comp:

print(\*line)

def РефлексивныеЗамыкания(m):

print("Рефлексивные замыкания, которые есть:")

print([(i, i) for i in range(len(m)) if m[i][i] == 1])

print("Рефлексивные замыкания, которых нет:")

print([(i,i) for i in range(len(m)) if m[i][i] == 0])

def ТранзитивныеЗамыкания1(m):

length = len(m)

tr\_1 = [[0 for \_ in range(0, length)] for \_ in range(0, length)]

for i in range(len(m)):

for j in range(len(m)):

for k in range(len(m)):

tr\_1[i][j] = (tr\_1[i][j] | (m[i][k] & m[k][j]))

print("Матрица tr\_1:")

for line in tr\_1:

print(\*line)

tr\_2 = [[0 for \_ in range(0, length)] for \_ in range(0, length)]

for i in range(len(m)):

for j in range(len(m)):

tr\_2[i][j] = m[i][j] | tr\_1[i][j]

print("Матрица tr\_2:")

for line in tr\_2:

print(\*line)

tr = [[0 for \_ in range(0, length)] for \_ in range(0, length)]

for i in range(len(m)):

for j in range(len(m)):

for k in range(len(m)):

tr[i][j] = (tr[i][j] | (tr\_2[i][k] & tr\_2[k][j]))

print("Матрица tr:")

for line in tr:

print(\*line)

def ТранзитивныеЗамыкания2(m):

def sum\_string(m, a, b):

n = len(m)

for i in range(n):

m[a][i] |= m[b][i]

return m

length = len(m)

pre\_arr = None

max\_count = 3

count = 0

while True:

for i in range(length):

for j in range(length):

if (i != j):

m = sum\_string(m, i, j)

if m == pre\_arr:

count += 1

else:

count = 0

pre\_arr = [a[:] for a in m]

if count >= max\_count:

return m

def пункт1():

clear()

global n

n = int(input('Введите мощность множества (1 <= n <= 100): '))

global R1

R1 = []

for i in range(n):

print('Введите ' + str(i + 1) + '-й элемент 1-го бинарного отношения (R1): ', end='')

R1.append(list(map(int, input().split())))

print()

global R2

R2 = []

for i in range(n):

print('Введите ' + str(i + 1) + '-й элемент 2-го бинарного отношения (R2): ', end='')

R2.append(list(map(int, input().split())))

ПечатьОтношений(R1, R2)

вназад()

def пункт2():

clear()

ПечатьОтношений(R1, R2)

if (Симметричность(R1)):

print("(R1) Симметричность (True) - 1")

else:

print("(R1) Ne Симметричность (False) - 0")

if (Симметричность(R2)):

print("(R2) Симметричность (True) - 1")

else:

print("(R2) Ne Симметричность (False) - 0")

вназад()

def пункт3():

clear()

ПечатьОтношений(R1, R2)

if (АнтиСимметричность(R1)):

print("(R1) АнтиСимметричность (True) - 1")

else:

print("(R1) Ne antiСимметричность (False) - 0")

if (АнтиСимметричность(R2)):

print("(R2) antiСимметричность (True) - 1")

else:

print("(R2) Ne antiСимметричность (False) - 0")

вназад()

def пункт4():

clear()

ПечатьОтношений(R1, R2)

if (Симетр(R1)):

print("(R1) Simmetry (True) - 1")

else:

print("(R1) Ne simmetry (False) - 0")

if (Симетр(R2)):

print("(R2) Simmetry (True) - 1")

else:

print("(R2) Ne simmetry (False) - 0")

вназад()

def пункт5():

clear()

ПечатьОтношений(R1, R2)

if (АнтиСиметр(R1)):

print("(R1) AntiSimmetry (True) - 1")

else:

print("(R1) Ne antiSimmetry (False) - 0")

if (АнтиСиметр(R2)):

print("(R2) AntiSimmetry (True) - 1")

else:

print("(R2) Ne antiSimmetry (False) - 0")

вназад()

def пункт6():

clear()

ПечатьОтношений(R1, R2)

if (Транзитивность(R1)):

print("(R1) Транзитивность (True) - 1")

else:

print("(R1) Ne transitivity (False) - 0")

if (Транзитивность(R2)):

print("(R2) Транзитивность (True) - 1")

else:

print("(R2) Ne transitivity (False) - 0")

вназад()

def пункт7():

clear()

ПечатьОтношений(R1, R2)

Композиция(R1, R2)

вназад()

def пункт8():

clear()

ПечатьОтношений(R1, R2)

print("Бинарное отношение R1:")

РефлексивныеЗамыкания(R1)

print("Бинарное отношение R2:")

РефлексивныеЗамыкания(R2)

вназад()

def пункт9():

clear()

ПечатьОтношений(R1, R2)

print("Транзитивное замыкание для R1 (Методом уножения и сложения матриц):")

ТранзитивныеЗамыкания1(R1)

print("Транзитивное замыкание для R2 (Методом умножения и сложения матриц):")

ТранзитивныеЗамыкания1(R2)

вназад()

def пункт10():

clear()

ПечатьОтношений(R1, R2)

print("Транзитивное замыкание для R1 (Методом Уоршолла):")

ТранзитивныеЗамыкания2(R1)

for line in R1:

print(\*line)

print("Транзитивное замыкание для R2 (Методом Уоршолла):")

ТранзитивныеЗамыкания2(R2)

for line in R2:

print(\*line)

вназад()

def menu():

print('Выберите пункт меню')

print(

'1. Создать бинарные отношения\n' + '2. Проверка отношений на рефлексивность\n' + '3. Проверка отношений на антирефлексивность\n' +

'4. Проверка отношений на симметричность\n' + '5. Проверка отношений на антисимметричность\n' + '6. Проверка отношений на транзитивность\n' +

'7. Композиция отношений\n' + '8. Рефлексивное замыкание\n' + '9. Транзитивное замыкание (Метод умножения и сложения матриц)\n' +

'10. Транзитивное замыкание (Метод Уоршолла)\n' + '0. Выход')

command = int(input('Введите номер пункта: '))

if command == 1:

пункт1()

elif command == 2:

пункт2()

elif command == 3:

пункт3()

elif command == 4:

пункт4()

elif command == 5:

пункт5()

elif command == 6:

пункт6()

elif command == 7:

пункт7()

elif command == 8:

пункт8()

elif command == 9:

пункт9()

elif command == 10:

пункт10()

elif command == 0:

print('Выход из программы!')

exit(0)

else:

print('Неверная команда')

def вназад():

choise = input('Чтобы вернуться назад нажмите "z"\n')

if choise == 'z':

clear()

menu()

else:

print('Неверная команда')

вназад()

if \_\_name\_\_ == '\_\_main\_\_':

menu()

Вывод: в ходе выполнения лабораторной работы были исследованы свойства бинарных отношений ,а также написана программа для определения свойств заданных бинарных отношений, построения композиции и нахождения замыкания.