|  |  |
| --- | --- |
| Gerb-BMSTU_01 | **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**

**«Замыкание бинарных отношений»**

**ДИСЦИПЛИНА: «Дискретная математика»**



|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Выполнил: студент гр. ИУК4-31Б | |  |  | ( | Суриков Н. С. | ) |
|  |  |  | (подпись) |  | (Ф.И.О.) |  |
| Проверил: | |  |  | ( | Никитенко У. В. | ) |
|  |  |  | (подпись) |  | (Ф.И.О.) |  |

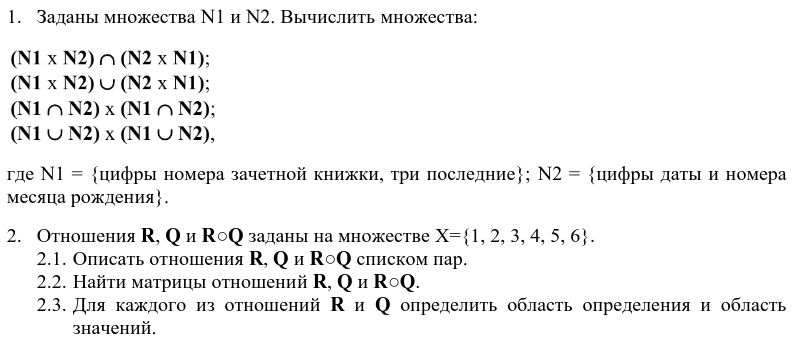
|  |  |
| --- | --- |
| Дата сдачи (защиты):  Результаты сдачи (защиты): | |
|  | - Балльная оценка:  - Оценка: |

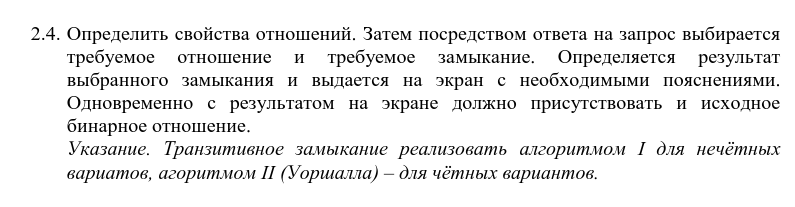
**Цель:** построение алгоритмов для определения свойств бинарного отношения и замыкания отношений.

**Задачи:**

1. разработать приложение, определяющее декартово произведение множеств, операции над отношениями, свойства бинарного отношения, заданного на конечном дискретном множестве, замыкание отношения.

**Вариант 10**

**

****

**Этапы выполнения:**

*Этап 1 — Выбор средств для решения задачи:*

Для решения задачи был выбран язык Python, а так же некоторые модули из numpy и tkinter, так как синтаксис языка достаточно прост, а библиотеки предоставляют готовые решения для быстрой разработки.

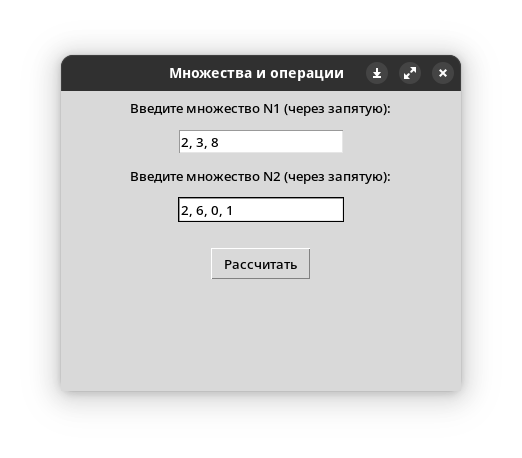
Установка: pip install numpy

*Этап 2 — Создание консольной версии программы*

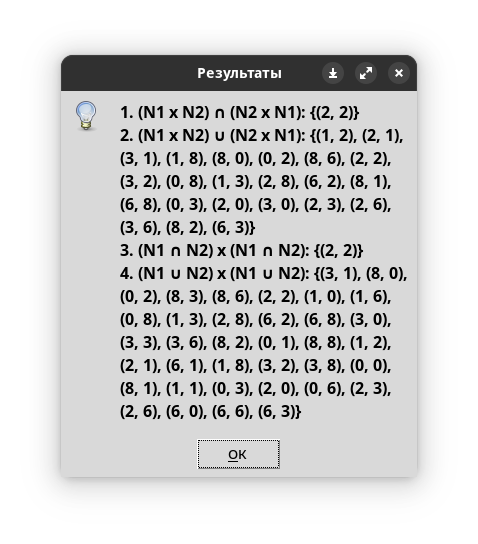
*Этап 3 — Создание графического интерфейса пользователя*

***Результат работы программы для задания 1:***

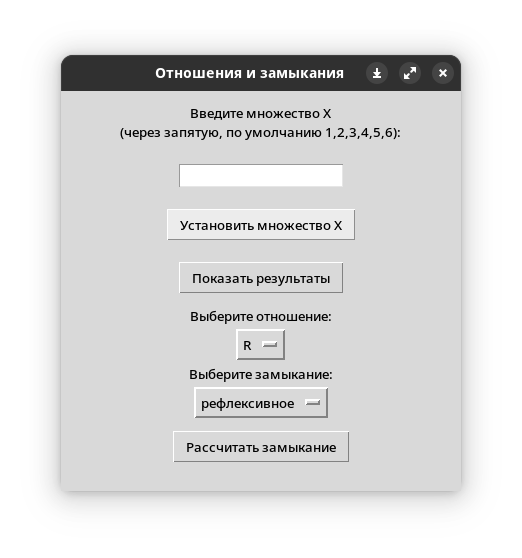
*Ввод:*

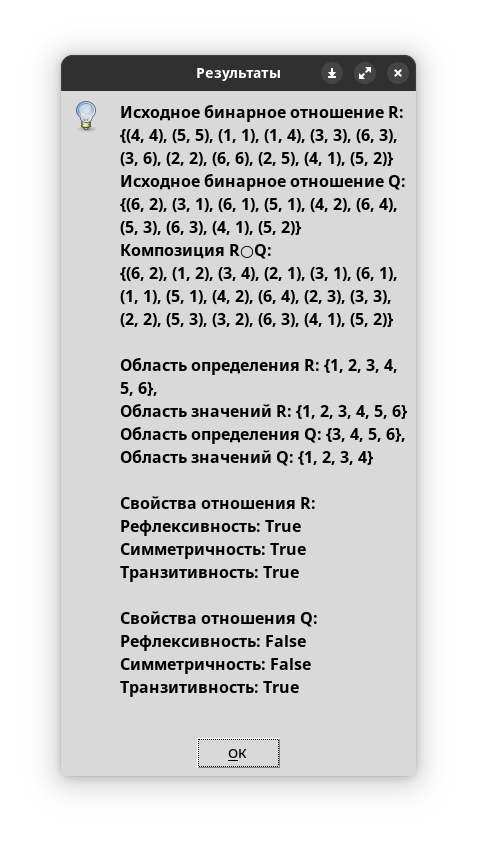
******

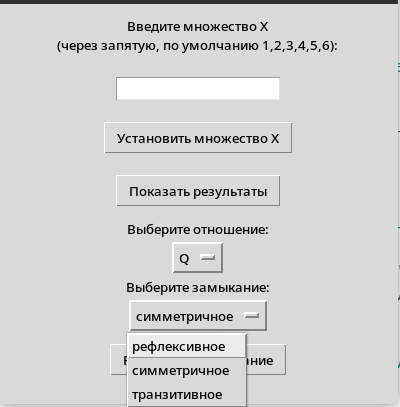
*Вывод:*

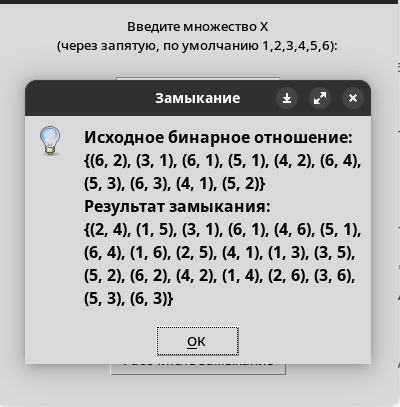
****

***Результат работы программы для задания 2:***

**

**

**

**

**Листинг программы:**

task1.py

1 import tkinter as tk

2 from tkinter import messagebox

3

4

5 def decart(s1, s2):

6 """Возвращает декартово произведение двух множеств."""

7 return set((a, b) for a in s1 for b in s2)

8

9

10 def calculate\_results():

11 """Выполняет расчеты и отображает результаты."""

12 try:

13 *# Получаем значения из полей ввода*

14 n1\_values = entry\_n1.get()

15 n2\_values = entry\_n2.get()

16

17 *# Преобразуем строки в множества*

18 N1 = set(map(int, n1\_values.split(",")))

19 N2 = set(map(int, n2\_values.split(",")))

20

21 N1\_x\_N2 = decart(N1, N2)

22 N2\_x\_N1 = decart(N2, N1)

23

24 inter = N1\_x\_N2 & N2\_x\_N1

25 union = N1\_x\_N2 | N2\_x\_N1

26

27 N1\_inter\_N2 = N1 & N2

28 inter\_dec = decart(N1\_inter\_N2, N1\_inter\_N2)

29

30 N1\_union\_N2 = N1 | N2

31 union\_dec = decart(N1\_union\_N2, N1\_union\_N2)

32

33 results = (

34 f"1. (N1 x N2) ∩ (N2 x N1): {inter}\n"

35 f"2. (N1 x N2) ∪ (N2 x N1): {union}\n"

36 f"3. (N1 ∩ N2) x (N1 ∩ N2): {inter\_dec}\n"

37 f"4. (N1 ∪ N2) x (N1 ∪ N2): {union\_dec}"

38 )

39

40 messagebox.showinfo("Результаты", results)

41

42 except ValueError:

43 messagebox.showerror(

44 "Ошибка", "Введите корректные числа, разделенные запятыми."

45 )

46

47

48 root = tk.Tk()

49 root.title("Множества и операции")

50 root.geometry("400x300")

51

52 label\_n1 = tk.Label(root, text="Введите множество N1 (через запятую):")

53 label\_n1.pack(pady=5)

54

55 entry\_n1 = tk.Entry(root)

56 entry\_n1.pack(pady=5)

57

58 label\_n2 = tk.Label(root, text="Введите множество N2 (через запятую):")

59 label\_n2.pack(pady=5)

60

61 entry\_n2 = tk.Entry(root)

62 entry\_n2.pack(pady=5)

63

64 calculate\_button = tk.Button(root, text="Рассчитать", command=calculate\_results)

65 calculate\_button.pack(pady=20)

66

67 root.mainloop()

task2.py

1 import tkinter as tk

2 from tkinter import messagebox

3 import numpy as np

4

5 default\_X = X = {1, 2, 3, 4, 5, 6}

6

7

8 def relation\_R(X):

9 """Возвращает бинарное отношение R для множества X."""

10 return {(m, n) for m in X for n in X if m % 3 == n % 3}

11

12

13 def relation\_Q(X):

14 """Возвращает бинарное отношение Q для множества X."""

15 return {(m, n) for m in X for n in X if (m - n) >= 2}

16

17

18 def composition\_R\_Q(R, Q):

19 """Возвращает композицию отношений R и Q."""

20 return {(m, p) for m, n1 in R for n2, p in Q if n1 == n2}

21

22

23 def relation\_matrix(X, relation):

24 """Возвращает матрицу отношения для заданного множества X и отношения."""

25 matrix = np.zeros((len(X), len(X)), dtype=int)

26 elements = list(X)

27 for m, n in relation:

28 matrix[elements.index(m)][elements.index(n)] = 1

29 return matrix

30

31

32 def domain\_and\_range(relation):

33 """Возвращает область определения и область значений для отношения."""

34 domain = {m for m, n in relation}

35 range\_ = {n for m, n in relation}

36 return domain, range\_

37

38

39 def is\_reflexive(X, relation):

40 """Проверяет, является ли отношение рефлексивным."""

41 return all((x, x) in relation for x in X)

42

43

44 def is\_symmetric(relation):

45 """Проверяет, является ли отношение симметричным."""

46 return all((n, m) in relation for m, n in relation)

47

48

49 def is\_transitive(relation):

50 """Проверяет, является ли отношение транзитивным."""

51 return all(

52 (m, p) in relation for m, n1 in relation for n2, p in relation if n1 == n2

53 )

54

55

56 def reflexive\_closure(X, relation):

57 """Возвращает рефлексивное замыкание отношения."""

58 return relation | {(x, x) for x in X}

59

60

61 def symmetric\_closure(relation):

62 """Возвращает симметричное замыкание отношения."""

63 return relation | {(n, m) for m, n in relation}

64

65

66 def transitive\_closure(X, relation):

67 """Возвращает транзитивное замыкание отношения."""

68 matrix = relation\_matrix(X, relation)

69 elements = list(X)

70 n = len(X)

71

72 for k in range(n):

73 for i in range(n):

74 for j in range(n):

75 matrix[i][j] = matrix[i][j] or (matrix[i][k] and matrix[k][j])

76

77 closure\_relation = set()

78 for i in range(n):

79 for j in range(n):

80 if matrix[i][j]:

81 closure\_relation.add((elements[i], elements[j]))

82 return closure\_relation

83

84

85 def show\_results():

86 """Отображает результаты отношений и их свойства."""

87 global X

88 if not X:

89 messagebox.showerror(

90 "Ошибка", "Пожалуйста, задайте множество X перед получением результатов."

91 )

92 return

93

94 R = relation\_R(X)

95 Q = relation\_Q(X)

96 R\_composed\_Q = composition\_R\_Q(R, Q)

97

98 results = f"Исходное бинарное отношение R:\n{R}\n"

99 results += f"Исходное бинарное отношение Q:\n{Q}\n"

100 results += f"Композиция R○Q:\n{R\_composed\_Q}\n\n"

101

102 domain\_R, range\_R = domain\_and\_range(R)

103 domain\_Q, range\_Q = domain\_and\_range(Q)

104

105 results += f"Область определения R: {domain\_R},\nОбласть значений R: {range\_R}\n"

106 results += f"Область определения Q: {domain\_Q},\nОбласть значений Q: {range\_Q}\n\n"

107

108 results += f"Свойства отношения R:\n"

109 results += f"Рефлексивность: {is\_reflexive(X, R)}\n"

110 results += f"Симметричность: {is\_symmetric(R)}\n"

111 results += f"Транзитивность: {is\_transitive(R)}\n\n"

112

113 results += f"Свойства отношения Q:\n"

114 results += f"Рефлексивность: {is\_reflexive(X, Q)}\n"

115 results += f"Симметричность: {is\_symmetric(Q)}\n"

116 results += f"Транзитивность: {is\_transitive(Q)}\n"

117

118 messagebox.showinfo("Результаты", results)

119

120

121 def calculate\_closure():

122 """Расчитывает замыкание выбранного отношения."""

123 closure\_type = closure\_var.get()

124 relation\_choice = relation\_var.get()

125

126 if not X:

127 messagebox.showerror(

128 "Ошибка", "Пожалуйста, задайте множество X перед расчетом замыкания."

129 )

130 return

131

132 R = relation\_R(X)

133 Q = relation\_Q(X)

134 R\_composed\_Q = composition\_R\_Q(R, Q)

135

136 if relation\_choice == "R":

137 selected\_relation = R

138 elif relation\_choice == "Q":

139 selected\_relation = Q

140 elif relation\_choice == "R○Q":

141 selected\_relation = R\_composed\_Q

142 else:

143 messagebox.showerror("Ошибка", "Неверный выбор отношения!")

144 return

145

146 if closure\_type == "рефлексивное":

147 closure = reflexive\_closure(X, selected\_relation)

148 elif closure\_type == "симметричное":

149 closure = symmetric\_closure(selected\_relation)

150 elif closure\_type == "транзитивное":

151 closure = transitive\_closure(X, selected\_relation)

152 else:

153 messagebox.showerror("Ошибка", "Неверный выбор замыкания!")

154 return

155

156 result\_message = f"Исходное бинарное отношение:\n{selected\_relation}\n"

157 result\_message += f"Результат замыкания:\n{closure}"

158

159 messagebox.showinfo("Замыкание", result\_message)

160

161

162 def set\_custom\_set():

163 """Устанавливает пользовательское множество X."""

164 global X

165 X = default\_X

166 input\_value = entry\_X.get()

167 if input\_value:

168 try:

169 X = set(map(int, input\_value.split(",")))

170 except ValueError:

171 messagebox.showerror(

172 "Ошибка", "Введите корректные числа, разделенные запятыми."

173 )

174 return

175

176

177 root = tk.Tk()

178 root.title("Отношения и замыкания")

179 root.geometry("400x400")

180

181 label\_X = tk.Label(

182 root, text="Введите множество X\n(через запятую, по умолчанию 1,2,3,4,5,6):"

183 )

184 label\_X.pack(pady=10)

185

186 entry\_X = tk.Entry(root)

187 entry\_X.pack(pady=10)

188

189 set\_button = tk.Button(root, text="Установить множество X", command=set\_custom\_set)

190 set\_button.pack(pady=10)

191

192 results\_button = tk.Button(root, text="Показать результаты", command=show\_results)

193 results\_button.pack(pady=10)

194

195 relation\_var = tk.StringVar(value="R")

196 relation\_label = tk.Label(root, text="Выберите отношение:")

197 relation\_label.pack()

198 relation\_options = ["R", "Q", "R○Q"]

199 relation\_menu = tk.OptionMenu(root, relation\_var, \*relation\_options)

200 relation\_menu.pack()

201

202 closure\_var = tk.StringVar(value="рефлексивное")

203 closure\_label = tk.Label(root, text="Выберите замыкание:")

204 closure\_label.pack()

205 closure\_options = ["рефлексивное", "симметричное", "транзитивное"]

206 closure\_menu = tk.OptionMenu(root, closure\_var, \*closure\_options)

207 closure\_menu.pack()

208

209 closure\_button = tk.Button(root, text="Рассчитать замыкание", command=calculate\_closure)

210 closure\_button.pack(pady=10)

211

212 root.mainloop()

213

**Вывод:** В ходе работы были изучены алгоритмы для определения свойств бинарного отношения и замыкания отношений. Разработаны программы для их расчета и вывода.