МИНОБРНАУКИ РОССИИ
САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ
ЭЛЕКТРОТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ
«ЛЭТИ» ИМ. В.И. УЛЬЯНОВА (ЛЕНИНА)
Кафедра ИБ

ОТЧЕТ

по лабораторной работе №3
по дисциплине «Модели безопасности компьютерных систем»
Тема: Модель Take-Grant

Студент гр. 8362 _____ Панфилович А.И.

Преподаватель _____ Савельев М.Ф.

Санкт-Петербург
2022

Задача.

Написать программу реализующую следующий функционал:

1. Обработку команд Grant, Remove, Create и фиксацию результатов в матрице доступа.

1.1. Формат функции Grant – grant(subj1, subj2, array[1,...,n])

1.2. Формат функции Remove – remove(subj1, array[1,...,n])

1.3. Формат функции Create – create(subj1, array[1,...,n])

2. Проверка корректности фиксации результатов проверяется с помощью приложения из 2ой лабораторной работы.

Теоретические сведения.

Классическая модель Take-Grant ориентирована на анализ путей распространения прав доступа в системах дискреционного разграничения доступа.

Основные элементы модели Take-Grant:

O – Множество объектов системы;

S – Множество субъектов системы;

R – Множество видов прав доступа, где t (take) – право брать право доступа; g (grant) – право давать права доступа;

G = (S, O, E) – конечный помеченный ориентированный граф без петель, представляющий текущие доступы в системе.

В классической модели Take-Grant рассматриваются четыре правила преобразования графа:

1) take() – брать права доступа;

2) grant() – давать права доступа;

3) create() – создавать новый объект или субъект; при этом субъект создатель-создатель может взять на созданный субъект любые права доступа (по умолчанию предполагается, что при выполнении правила create() создается объект);

4) remove() – удалять права доступа.

Разработка программы.

В ходе выполнения задания было разработано GUI приложение для пользователя, позволяющее обрабатывать введенную пользователем строку (команду), в случае ее корректности программа выполняет ее и вносит изменения в матрицу доступа в таблице Excel (см. рис. 1). Интерфейс программы изображен на рисунке 2.

	у	е	ы	а	о	э	я	и	ю
user1	0	1	1	1	1	1	0	0	0
user2	0	0	0	0	0	0	0	0	0
user3	0	0	0	0	0	0	0	0	0
user4	0	0	0	0	0	0	0	0	0

Рисунок 1 – Матрица доступа в таблице Excel

Интерфейс программы содержит поле для ввода команд (сверху окна) в формате $command(\mathrm{arg}_1, \mathrm{arg}_2, \ldots, \mathrm{arg}_n)$. Одновременно в окне можно задавать только одну команду. Нажатием кнопки команда выполняется и результат ее обработки выводится в нижнем поле программы (там выводятся сообщения об успешном выполнении и сообщения об ошибках).

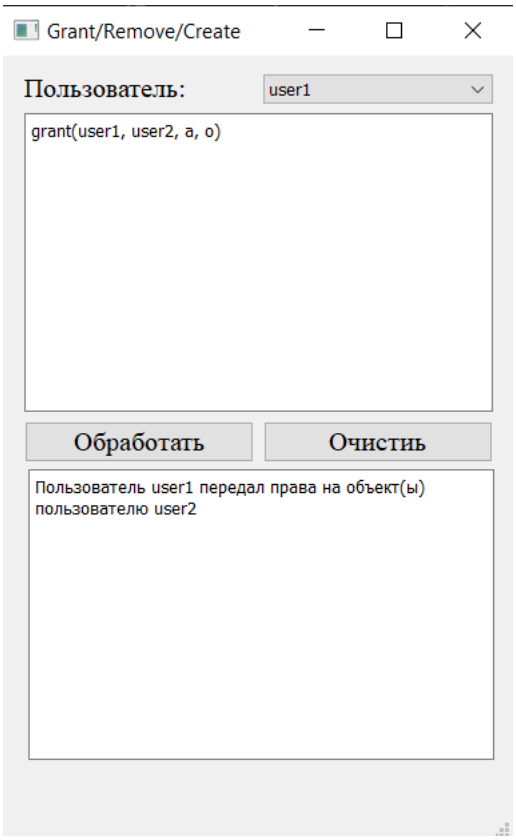

Рисунок 2 – Интерфейс программы пользователя

Пример работы программы.

Выполним несколько команд чтобы продемонстрировать работу программы, зафиксируем состояния матрицы доступа и проверим результаты программой из второй лабораторной работы.

Изначальное состояние матрицы доступа:

	у	е	ы	а	о	э	я	и	ю
user1	0	1	1	1	1	1	0	0	0
user2	0	0	0	1	1	0	0	0	0
user3	0	0	0	0	0	0	0	0	0
user4	0	0	0	0	0	0	0	0	0

Рисунок 3 – Изначальное состояние матрицы доступа

1. Пример выполнения команды grant:

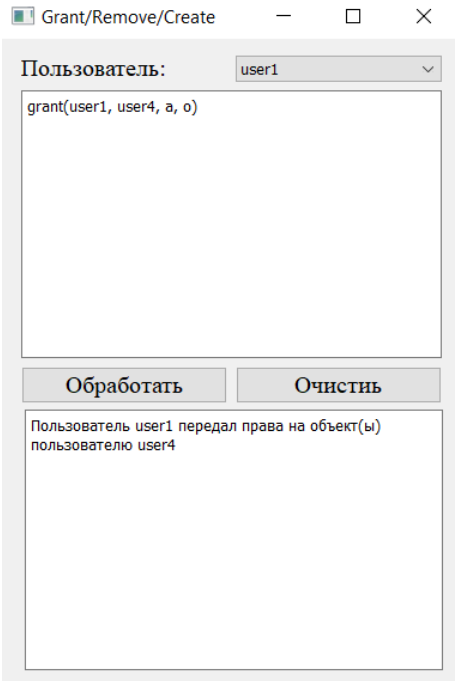

Рисунок 4 – Выполнение команды grant

Состояние матрицы доступа после выполнения программы:

	у	е	ы	а	о	э	я	и	ю
user1	0	1	1	1	1	1	0	0	0
user2	0	0	0	1	1	0	0	0	0
user3	0	0	0	0	0	0	0	0	0
user4	0	0	0	1	1	0	0	0	0

Рисунок 5 – Матрица доступа после применения grant

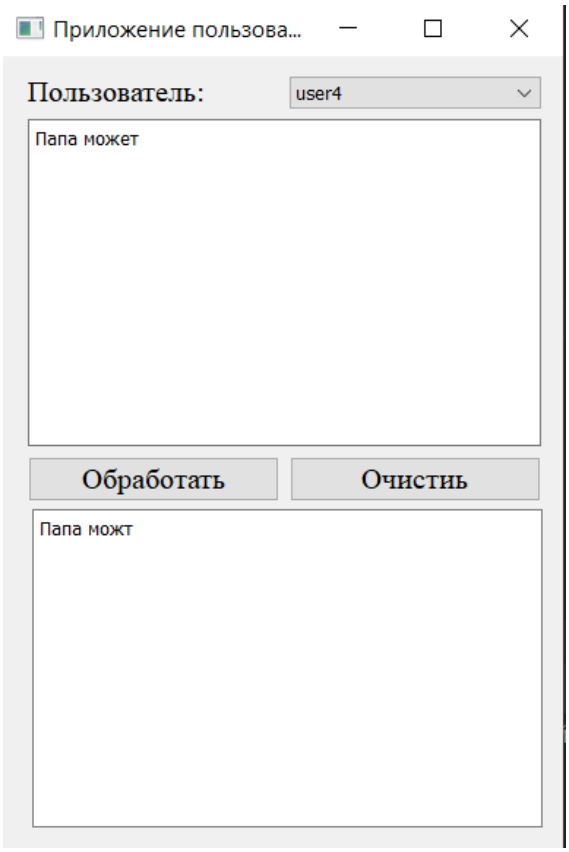

Рисунок 6 – Проверка результатов

2. Пример выполнения команды remove:

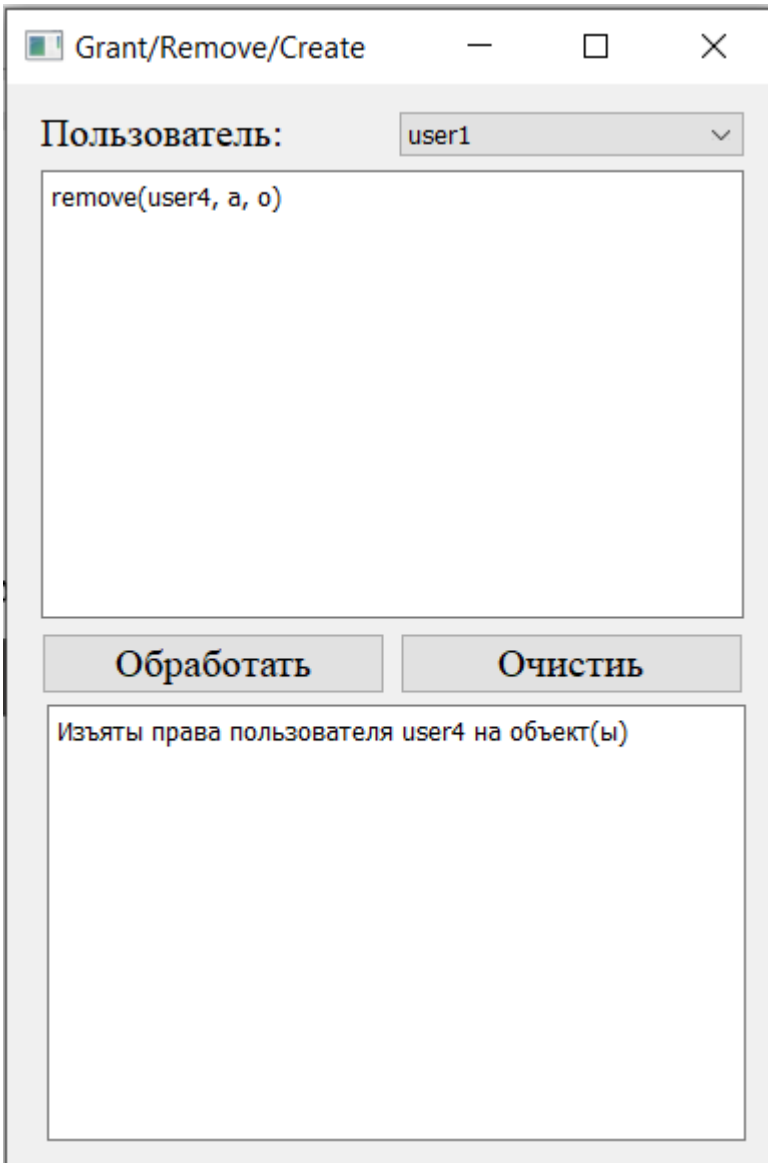

Рисунок 7 – Выполнение команды remove

	у	е	ы	а	о	э	я	и	ю
user1	0	1	1	1	1	1	0	0	0
user2	0	0	0	1	1	0	0	0	0
user3	0	0	0	0	0	0	0	0	0
user4	0	0	0	0	0	0	0	0	0

Рисунок 8 – Матрица доступа после применения remove

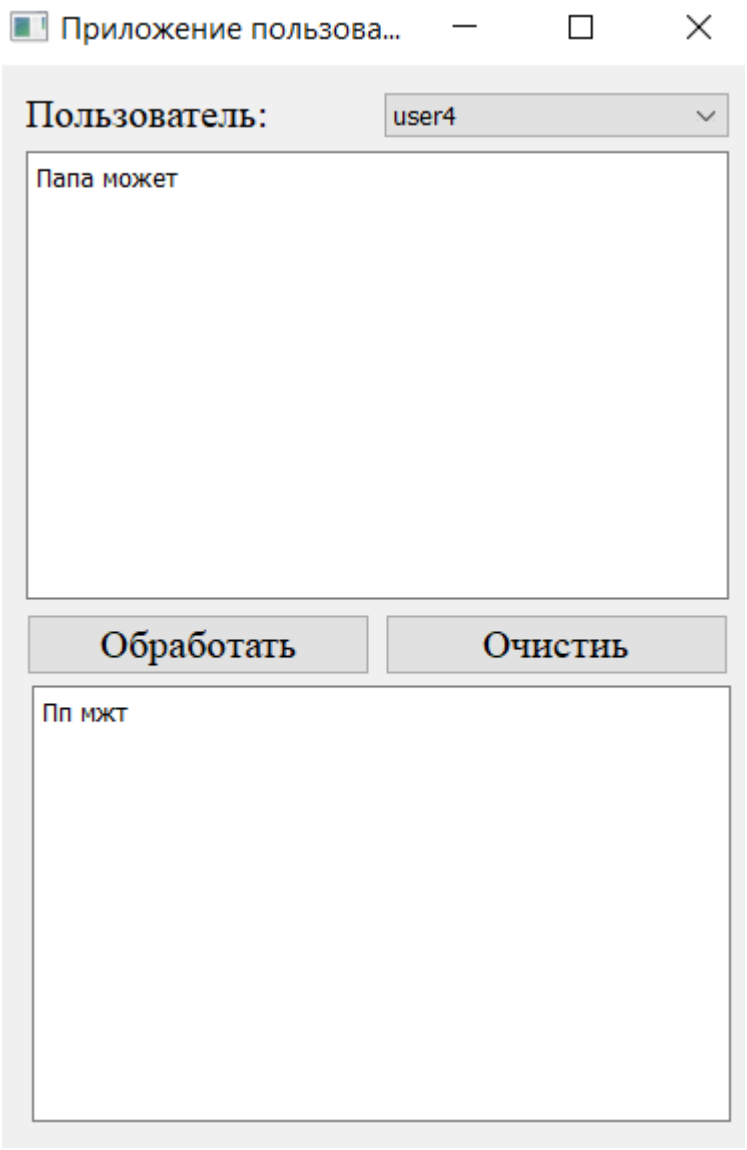

Рисунок 9 – Проверка результатов

3. Пример выполнения команды create:

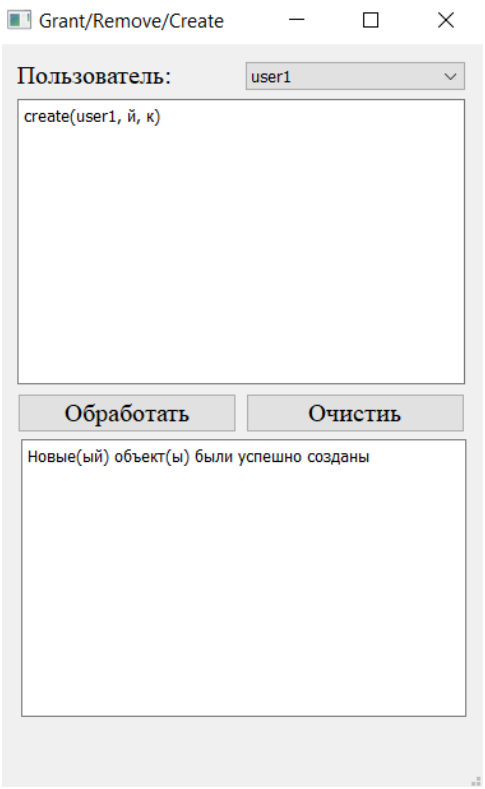

Рисунок 10 – Выполнение команды create

	у	е	ы	а	о	э	я	и	ю	й	к
user1	0	1	1	1	1	1	0	0	0	1	1
user2	0	0	0	1	1	0	0	0	0	0	0
user3	0	0	0	0	0	0	0	0	0	0	0
user4	0	0	0	0	0	0	0	0	0	0	0

Рисунок 11 – Матрица доступа после применения create

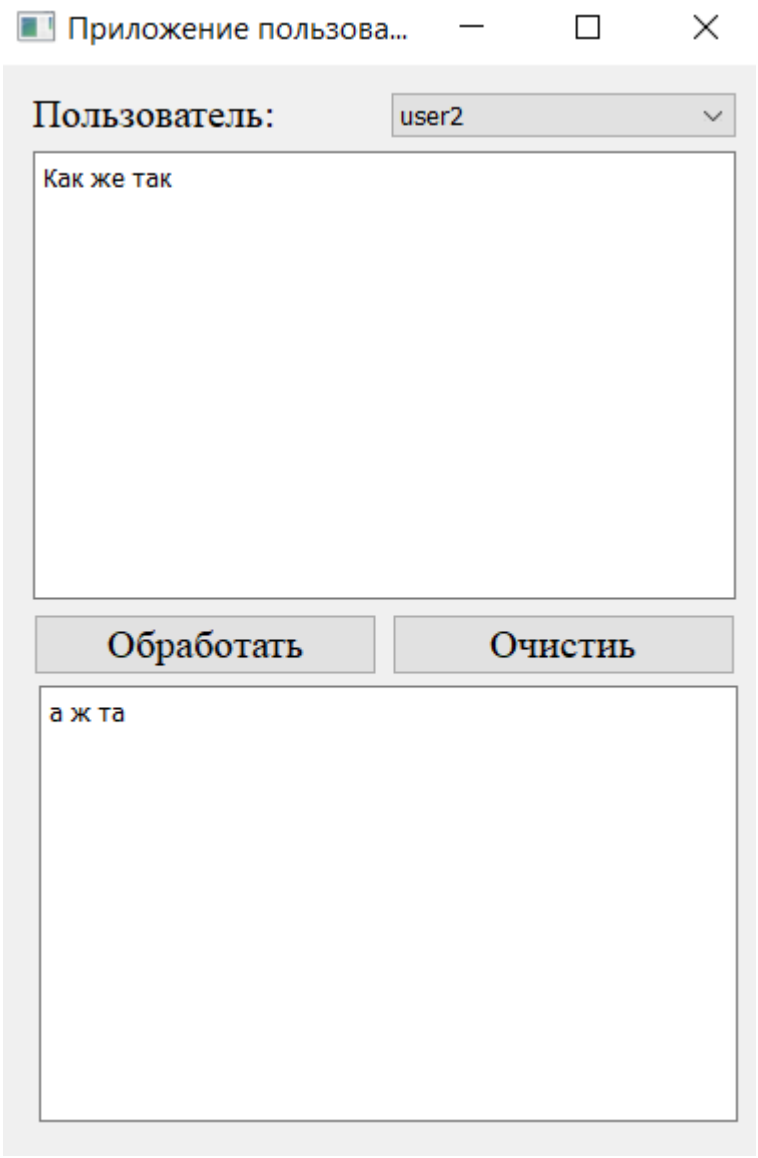

Рисунок 12 – Проверка результатов

Выводы.

В результате выполнения лабораторной работы было разработано приложение на языке Python 3.8, GUI приложение пользователя также использовало библиотеку PyQt5. Это приложение эмулируют работу пользователей в системе с дискреционным разграничением доступа и применение ими правил классической модели Take-Grant для распространения прав доступа в системе.

ПРИЛОЖЕНИЕ А

Исходный код программы

Файл main.py:

```
from PyQt5 import QtCore, QtGui, QtWidgets
import sys, openpyxl
import commands

class Ui_MainWindow(object):

    users = []
    users_name = []
    litters = []
    sheet = 0

    def __init__(self):
        file = "rules.xlsx"
        wb = openpyxl.load_workbook(file)
        self.sheet = wb['rule']
        self.users = self.sheet['A'][1:]
        self.users_name = [name.value for name in self.users]
        self.litters = []


    def setupUi(self, MainWindow):
        MainWindow.setObjectName("MainWindow")
        MainWindow.resize(383, 588)
        self.centralwidget = QtWidgets.QWidget(MainWindow)
        self.centralwidget.setObjectName("centralwidget")
        self.layoutWidget = QtWidgets.QWidget(self.centralwidget)
        self.layoutWidget.setGeometry(QtCore.QRect(17, 14, 351, 291))
        self.layoutWidget.setObjectName("layoutWidget")
        self.verticalLayout = QtWidgets.QVBoxLayout(self.layoutWidget)
        self.verticalLayout.setContentsMargins(0, 0, 0, 0)
        self.verticalLayout.setObjectName("verticalLayout")
        self.horizontalLayout = QtWidgets.QHBoxLayout()
        self.horizontalLayout.setObjectName("horizontalLayout")
        self.label = QtWidgets.QLabel(self.layoutWidget)
        font = QtGui.QFont()
        font.setFamily("Times New Roman")
        font.setPointSize(12)
        self.label.setFont(font)
        self.label.setObjectName("label")
        self.horizontalLayout.addWidget(self.label)
        self.names_comboBox = QtWidgets.QComboBox(self.layoutWidget)
        self.names_comboBox.setObjectName("names_comboBox")
        self.names_comboBox.addItems(self.users_name)

        self.horizontalLayout.addWidget(self.names_comboBox)
```

```
        self.verticalLayout.addLayout(self.horizontalLayout)
        self.Start_text = QtWidgets.QTextEdit(self.layoutWidget)
        self.Start_text.setObjectName("Start_text")
        self.verticalLayout.addWidget(self.Start_text)
        self.horizontalLayout_2 = QtWidgets.QHBoxLayout()
        self.horizontalLayout_2.setObjectName("horizontalLayout_2")
        self.start_btn = QtWidgets.QPushButton(self.layoutWidget)
        font = QtGui.QFont()
        font.setFamily("Times New Roman")
        font.setPointSize(12)
        self.start_btn.setFont(font)
        self.start_btn.setObjectName("start_btn")
        self.horizontalLayout_2.addWidget(self.start_btn)
        self.clear_btn = QtWidgets.QPushButton(self.layoutWidget)
        font = QtGui.QFont()
        font.setFamily("Times New Roman")
        font.setPointSize(12)
        self.clear_btn.setFont(font)
        self.clear_btn.setObjectName("clear_btn")
        self.horizontalLayout_2.addWidget(self.clear_btn)
        self.verticalLayout.addLayout(self.horizontalLayout_2)
        self.New_text = QtWidgets.QTextEdit(self.centralwidget)
        self.New_text.setGeometry(QtCore.QRect(20, 310, 349, 218))
        self.New_text.setObjectName("New_text")
        MainWindow.setCentralWidget(self.centralwidget)
        self.menubar = QtWidgets.QMenuBar(MainWindow)
        self.menubar.setGeometry(QtCore.QRect(0, 0, 383, 26))
        self.menubar.setObjectName("menubar")
        MainWindow.setMenuBar(self.menubar)
        self.statusbar = QtWidgets.QStatusBar(MainWindow)
        self.statusbar.setObjectName("statusbar")
        MainWindow.setStatusBar(self.statusbar)

        self.retranslateUi(MainWindow)
        QtCore.QMetaObject.connectSlotsByName(MainWindow)

        self.add_function()

    def add_function(self):
        self.clear_btn.clicked.connect(lambda: self.clear())
        self.start_btn.clicked.connect(lambda: self.start())

    def clear(self):
        self.Start_text.setText("")
        self.New_text.setText("")

    def start(self):
        # read input
        command = [s.strip() for s in self.Start_text.toPlainText().split('(')]
```

```
        # parse command
        output = ''
        mod = command[0].lower()

        if command[0] and command[1] and command[1][-1] == ')':
            args = [s.strip() for s in command[1][:-1].split(',')]
            if len(args) > 1:
                if mod == 'grant':
                    output = commands.grant(args)
                elif mod == 'remove':
                    output = commands.remove(args)
                elif mod == 'create':
                    output = commands.create(args)
                else:
                    output = 'Команда не была распознана!'
            else:
                output = 'Команда содержит недостаточное кол-во аргументов!'
        else:
            output = 'Команда содержит синтаксическую ошибку!'

        self.New_text.setText(output)

    def retranslateUi(self, MainWindow):
        _translate = QtCore.QCoreApplication.translate
        MainWindow.setWindowTitle(_translate("MainWindow", "MainWindow"))
        self.label.setText(_translate("MainWindow", "Пользователь:"))
        self.start_btn.setText(_translate("MainWindow", "Обработать"))
        self.clear_btn.setText(_translate("MainWindow", "Очистить"))


def main():
    app = QtWidgets.QApplication(sys.argv)
    MainWindow = QtWidgets.QMainWindow()
    ui = Ui_MainWindow()
    ui.setupUi(MainWindow)
    MainWindow.show()
    sys.exit(app.exec_())

main()
```

Файл commands.py:

```
from pathlib import Path
import openpyxl as xl


def grant(args):
    # open excel file
```

```
    path = Path(__file__).parents[1]
    excel_file = path / 'rules.xlsx'
    wb = xl.load_workbook(excel_file)
    rules = wb.active

    # args check
    output = ''
    if len(args) < 3:
        output = 'Команда Grant должна содержать не менее 3 аргументов!'
    else:
        users = [u.value for u in rules['A'][1:]]
        symbols = [s.value for s in rules['1'][1:]]
        if args[0] in users and args[1] in users:
            for arg in args[2:]:
                if arg not in symbols:
                    output = 'Команда содержит несуществующий символ!'
                    break
        else:
            output = 'Команда содержит несуществующего пользователя!'

        if not output:
            subj1 = None
            subj2 = None
            for u in rules['A'][1:]:
                if u.value == args[0]:
                    subj1 = u
                if u.value == args[1]:
                    subj2 = u

            for arg in args[2:]:
                for s in rules['1'][1:]:
                    if s.value == arg:
                        if rules[subj1.row][s.column-1].value == 1:
                            rules[subj2.row][s.column-1].value = 1
                        else:
                            output = 'Пользователь не может передать права на один из объектов!'
                            return output
            output = 'Пользователь %s передал права на объект(ы) пользователю %s' % (subj1.value,
subj2.value)
    wb.save(excel_file)
    return output


def remove(args):
    # open excel file
    path = Path(__file__).parents[1]
    excel_file = path / 'rules.xlsx'
    wb = xl.load_workbook(excel_file)
    rules = wb.active
```

```
    # args check
    output = ''
    users = [u.value for u in rules['A'][1:]]
    symbols = [s.value for s in rules['1'][1:]]
    if args[0] in users:
        for arg in args[1:]:
            if arg not in symbols:
                output = 'Команда содержит несуществующий символ!'
                break
    else:
        output = 'Команда содержит несуществующего пользователя!'

    if not output:
        for u in rules['A'][1:]:
            if u.value == args[0]:
                for arg in args[1:]:
                    for s in rules['1'][1:]:
                        if s.value == arg:
                            rules[u.row][s.column-1].value = 0
                            output = 'Изъяты права пользователя %s на объект(ы)' % u.value
    wb.save(excel_file)
    return output


def create(args):
    # open excel file
    path = Path(__file__).parents[1]
    excel_file = path / 'rules.xlsx'
    wb = xl.load_workbook(excel_file)
    rules = wb.active
    output = ''

    # args check
    users = [u.value for u in rules['A'][1:]]
    symbols = [s.value for s in rules['1'][1:]]
    if args[0] in users:
        for arg in args[1:]:
            if len(arg) == 1:
                if arg in symbols:
                    output = 'Команда Create не может давать права на существующий символ!'
                    break
            else:
                output = 'Команда содержит непраильный символ!'
                break
    else:
        output = 'Команда содержит несуществующего пользователя!'

    if not output:
        for u in rules['A'][1:]:
            if u.value == args[0]:
```

```
            for arg in args[1:]:
                col = rules['1'][-1].column + 1
                last = rules.cell(row=1, column=col, value=arg)
                for cell in rules[last.column_letter][1:]:
                    if cell.row == u.row:
                        cell.value = 1
                    else:
                        cell.value = 0
    output = 'Новые(ый) объект(ы) были успешно созданы'
wb.save(excel_file)
return output
```