

Министерство науки и высшего образования РФ
Федеральное государственное автономное образовательное
учреждение
высшего образования
«Московский политехнический университет»
(Московский политех)

Отчёт по курсу «Программирование криптографических алгоритмов»
Лабораторная работа 1.3. Шифры блочной замены



Выполнил:

Студент группы 221-352

Иванов В. В.

Проверил преподаватель: Бутакова Н. Г.

Москва 2024г.

Аннотация

- **Среда программирования**

- Visual Studio Code

- **Язык программирования**

- Python

- **Процедуры для запуска программы**

- Visual Studio Code (main.py)

- **Пословица-тест**

- Тот, кто ложится на два стула, падает на ребра.

- **Текст для проверки работы (не меньше 1000 знаков (1430))**

Жизнь - это удивительное приключение, полное разнообразных событий и встреч. В каждом моменте мы находим что-то новое и уникальное. Стремление к росту и саморазвитию вдохновляет нас на поиск новых горизонтов. Важно помнить, что каждый шаг вперед приносит с собой уроки и опыт.

Разнообразие культур, языков и традиций делает наш мир удивительно богатым. Общение с людьми разных национальностей расширяет кругозор, позволяя нам понимать и уважать друг друга. Взаимное уважение и терпимость создают основу для гармоничного сосуществования.

Природа тоже играет важную роль в нашей жизни. Красота закатов, шум океана, пение птиц - все это напоминает нам о величии мира природы. Забота о окружающей среде становится неотъемлемой частью ответственного образа жизни.

Работа и творчество придают смысл нашим усилиям. Стремление к достижению целей мотивирует нас на новые начинания. Каждый проект, даже самый маленький, приносит удовлетворение и чувство выполненного долга.

Семья и друзья являются надежной опорой в нашей жизни. Обмен историями, веселые посиделки и поддержка в трудные моменты создают теплую атмосферу взаимопонимания и любви.

Таким образом, наша жизнь - это мозаика различных моментов, соединенных воедино. Важно ценить каждый момент и стремиться делать мир вокруг нас ярче и лучше. С любовью, терпением и целеустремленностью мы можем создавать свою уникальную историю, наполненную смыслом и радостью.

• Код программы-интерфейса

```
• import sys
• import random
• from PyQt5.QtWidgets import QApplication, QWidget, QVBoxLayout, QHBoxLayout, QLabel, QLineEdit, QPushButton, QComboBox,
  QTextEdit, QCheckBox
• from PyQt5.QtCore import Qt
• from atbash import atbash_encrypt, atbash_decrypt
• from cesar import cesar_encrypt, cesar_decrypt, cesar_check_parameters
• from polibia import polibia_encrypt, polibia_decrypt
• from tritemiy import tritemiy_encrypt, tritemiy_decrypt
• from belazo import belazo_encrypt, belazo_decrypt, belazo_check_parameters
• from vigenere import vigenere_encrypt, vigenere_decrypt, vigenere_check_parameters
• from S_block import s_block_encrypt, s_block_decrypt
• from matrix import matrix_encrypt, matrix_decrypt, matrix_check_parameters, multiply_matrix, determinant,
  adjugate_matrix, inverse_matrix
• from playfair import playfair_encrypt, playfair_decrypt, playfair_check_parameters
• # from verticalTransposition import vertical_transposition_encrypt, vertical_transposition_decrypt,
  vertical_transposition_check_parameters
•
• available_ciphers = [
•     "Шифр АТБАШ", "Шифр Цезаря", "Шифр Полибия",
•     "Шифр Тритемия", "Шифр Белазо", "Шифр Виженера", "МАГМА(s_block)",
•     "Шифр Матричный", "Шифр Плейфера", # "Вертикальная Транспозиция",
• ]
•
• alphabet = [
•     "а", "б", "в", "г", "д", "е", "ж", "з", "и", "й", "к", "л", "м",
•     "н", "о", "п", "р", "с", "т", "у", "ф", "х", "ц", "ч", "ш", "щ",
•     "ъ", "ы", "ь", "э", "ю", "я"
• ]
•
• alphabet_polibia = [
```

```

•     ["а", "б", "в", "г", "д", "е"],
•     ["ж", "з", "и", "й", "к", "л"],
•     ["м", "н", "о", "п", "р", "с"],
•     ["т", "у", "ф", "х", "ц", "ч"],
•     ["ш", "щ", "ъ", "ы", "ь", "э"],
•     ["ю", "я"]
• ]
•
• alphabet_playfair = [
•     "а", "б", "в", "г", "д", "е", "ж", "з", "и", "к", "л", "м", "н",
•     "о", "п", "р", "с", "т", "у", "ф", "х", "ц", "ч", "ш", "щ", "ъ",
•     "ы", "э", "ю", "я"
• ]
•
• alphabet_sblock = ["0", "1", "2", "3", "4", "5", "6", "7", "8", "9", "a", "b", "c", "d", "e", "f"]
•
• mem = {
•     "bigTextFlag": False,
•     "vigenereSwitch": False,
•     "mode": "encrypt",
• }
•
• class CipherApp(QWidget):
•     def __init__(self):
•         super().__init__()
•         self.initUI()
•
•     def initUI(self):
•         self.setWindowTitle('Шифры')
•         self.resize(960, 640)
•         layout = QVBoxLayout()

```

```
• # Выбор шифра
• cipher_layout = QHBoxLayout()
• cipher_label = QLabel('Выберите шифр:')
• self.cipher_combo = QComboBox()
• self.cipher_combo.addItem(available_ciphers)
• cipher_layout.addWidget(cipher_label)
• cipher_layout.addWidget(self.cipher_combo)
•
• # Ввод открытого текста
• open_text_label = QLabel('Введите открытый текст(Расшифрованный):')
• self.open_text_edit = QTextEdit()
•
• # Ввод зашифрованного текста
• cipher_text_label = QLabel('Шифрованный текст:')
• self.cipher_text_edit = QTextEdit()
•
• # Ввод сдвига для шифра Цезаря
• self.cesar_shift_edit = QLineEdit()
• self.cesar_shift_edit.setPlaceholderText('Введите сдвиг для шифра Цезаря')
• self.cesar_shift_edit.textChanged.connect(self.check_cesar_shift)
•
• # Ввод ключевого слова для шифра Белазо или Плейфера
• self.keyword_edit = QLineEdit()
• self.keyword_edit.setPlaceholderText('Введите ключевое слово для шифра Белазо или Плейфера')
•
• # Ввод ключевой буквы для шифра Виженера
• self.vigener_key_edit = QLineEdit()
• self.vigener_key_edit.setPlaceholderText('Введите ключевую букву для шифра Виженера')
• self.vigener_key_edit.textChanged.connect(self.check_vigener_key)
•
• # Ввод ключевой матрицы для шифра Матричный
• self.matrix_edit = QLineEdit()
```

```

• self.matrix_edit.setPlaceholderText('Введите ключевую матрицу для шифра Матричный')
•
• # Ввод ключа для шифра вертикальной транспозиции
• # self.vertical_transposition_keyword_edit = QLineEdit()
• # self.vertical_transposition_keyword_edit.setPlaceholderText('Введите ключ для шифра вертикальной
транспозиции')
•
• # Режим работы шифра (шифрование или дешифрование)
• mode_layout = QHBoxLayout()
• mode_label = QLabel('Выберите режим:')
• self.mode_combo = QComboBox()
• self.mode_combo.addItem('Шифрование', 'Расшифрование'])
• mode_layout.addWidget(mode_label)
• mode_layout.addWidget(self.mode_combo)
•
• # Кнопка для запуска шифрования/дешифрования
• self.encrypt_button = QPushButton('Выполнить')
•
• layout.addLayout(cipher_layout)
• layout.addWidget(open_text_label)
• layout.addWidget(self.open_text_edit)
• layout.addWidget(cipher_text_label)
• layout.addWidget(self.cipher_text_edit)
• layout.addWidget(self.cesar_shift_edit)
• layout.addWidget(self.keyword_edit)
• layout.addWidget(self.vigener_key_edit)
• layout.addWidget(self.matrix_edit)
• # layout.addWidget(self.vertical_transposition_keyword_edit)
• layout.addLayout(mode_layout)
• layout.addWidget(self.encrypt_button)
•
• self.setLayout(layout)

```

```

•
•     # Переключатель для выбора режима текста
•     self.text_mode_checkbox = QCheckBox('Расширенный текст')
•     layout.addWidget(self.text_mode_checkbox)
•
•     # Подключение слотов к сигналам
•     self.encrypt_button.clicked.connect(self.cipher_parser)
•     self.text_mode_checkbox.stateChanged.connect(self.handle_text_mode_change)
•
• def handle_text_mode_change(self, state):
•     if state == Qt.Checked:
•         mem["bigTextFlag"] = True
•     else:
•         mem["bigTextFlag"] = False
•
• def check_cesar_shift(self):
•     shift_text = self.cesar_shift_edit.text()
•     try:
•         shift = int(shift_text)
•         if shift < 0 or shift >= len(alphabet):
•             self.cesar_shift_edit.setStyleSheet("QLineEdit { color: red; }")
•         else:
•             self.cesar_shift_edit.setStyleSheet("")
•     except ValueError:
•         self.cesar_shift_edit.setStyleSheet("QLineEdit { color: red; }")
•
• def check_vigener_key(self):
•     key_text = self.vigener_key_edit.text()
•     if len(key_text) != 1 or key_text.lower() not in alphabet:
•         self.vigener_key_edit.setStyleSheet("QLineEdit { color: red; }")
•     else:
•         self.vigener_key_edit.setStyleSheet("")

```

```

def text_preparation(self, text):
    bigTextFlag = mem["bigTextFlag"]
    if bigTextFlag:
        # Обработка расширенного текста
        return text.replace("ё", "е").replace(".", "тчк").replace(",", "зпт").replace("-", "тире").replace(" ",
"прбл").replace(":", "двтч").replace(";", "тчсзп").replace("(", "отскб").replace(")", "зксб").replace("?",
"впрзн").replace("!", "восклзн").replace("\n", "првст").lower()
    else:
        # Обработка обычного текста
        return text.replace("ё", "е").replace(".", "тчк").replace(",", "зпт").replace("-", "тире").replace(" ",
"" ).replace(":", "").replace(";", "").replace("(", "").replace(")", "").replace("?", "").replace("!", "").replace("\n",
"" ).lower()

def cipher_parser(self):
    cipher_choose_input = self.cipher_combo.currentText()
    open_text_input = self.open_text_edit.toPlainText()
    cipher_text_input = self.cipher_text_edit.toPlainText()
    cesar_shift = self.cesar_shift_edit.text()
    keyword = self.keyword_edit.text()
    vigenere_keyletter = self.vigenere_key_edit.text()
    matrix_input = self.matrix_edit.text()

    # Определение режима работы (шифрование или дешифрование)
    mode = 'encrypt' if self.mode_combo.currentText() == 'Шифрование' else 'decrypt'

    # Определение флага для обработки больших текстов
    bigTextFlag = len(open_text_input) > 1000 #ваш порог длины текста

    if cipher_choose_input == "Шифр АТБАШ":
        if mode == "encrypt":
            cipher_text_input = atbash_encrypt(self.text_preparation(open_text_input), alphabet)

```



```

•         elif mode == "decrypt":
•             open_text_input = atbash_decrypt(cipher_text_input, alphabet)
•     elif cipher_choose_input == "Шифр Цезаря":
•         if cesar_shift: # Проверка на пустую строку
•             cesar_shift = int(cesar_shift)
•             if cesar_check_parameters(cesar_shift, alphabet):
•                 if mode == "encrypt":
•                     cipher_text_input = cesar_encrypt(self.text_preparation(open_text_input), cesar_shift,
alphabet)
•                 elif mode == "decrypt":
•                     open_text_input = cesar_decrypt(cipher_text_input, cesar_shift, alphabet)
•             else:
•                 if mode == "encrypt":
•                     cipher_text_input = "Проверьте правильность ввода сдвига"
•                 elif mode == "decrypt":
•                     open_text_input = "Проверьте правильность ввода сдвига"
•             else:
•                 if mode == "encrypt":
•                     cipher_text_input = "Введите сдвиг для шифра Цезаря"
•                 elif mode == "decrypt":
•                     open_text_input = "Введите сдвиг для шифра Цезаря"
•     elif cipher_choose_input == "Шифр Полибия":
•         if mode == "encrypt":
•             cipher_text_input = polibia_encrypt(self.text_preparation(open_text_input), alphabet_polibia)
•         elif mode == "decrypt":
•             open_text_input = polibia_decrypt(cipher_text_input, alphabet_polibia)
•     elif cipher_choose_input == "Шифр Тритемия":
•         if mode == "encrypt":
•             cipher_text_input = tritemiy_encrypt(self.text_preparation(open_text_input), alphabet)
•         elif mode == "decrypt":
•             open_text_input = tritemiy_decrypt(cipher_text_input, alphabet)
•     elif cipher_choose_input == "Шифр Белазо":

```

```

•         if keyword:
•             if belazo_check_parameters(keyword.lower(), alphabet):
•                 if mode == "encrypt":
•                     cipher_text_input = belazo_encrypt(self.text_preparation(open_text_input), keyword.lower(),
alphabet)
•
•                     elif mode == "decrypt":
•                         open_text_input = belazo_decrypt(cipher_text_input, keyword.lower(), alphabet)
•
•             else:
•                 if mode == "encrypt":
•                     cipher_text_input = "Проверьте правильность ввода ключевого слова"
•                 elif mode == "decrypt":
•                     open_text_input = "Проверьте правильность ввода ключевого слова"
•
•             else:
•                 if mode == "encrypt":
•                     cipher_text_input = "Введите ключевое слово для шифра Белазо"
•                 elif mode == "decrypt":
•                     open_text_input = "Введите ключевое слово для шифра Белазо"
•
•         elif cipher_choose_input == "Шифр Виженера":
•             if vigener_keyletter:
•                 if vigener_check_parameters(vigener_keyletter, alphabet):
•                     mode = "encrypt" if self.mode_combo.currentText() == 'Шифрование' else 'decrypt'
•                     if mode == "encrypt":
•                         cipher_text_input = vigener_encrypt(self.text_preparation(open_text_input), vigener_keyletter,
"selfkey", alphabet)
•                     elif mode == "decrypt":
•                         open_text_input = vigener_decrypt(cipher_text_input, vigener_keyletter, "selfkey", alphabet)
•
•             else:
•                 if mode == "encrypt":
•                     cipher_text_input = "Проверьте правильность ввода ключевой буквы"
•                 elif mode == "decrypt":
•                     open_text_input = "Проверьте правильность ввода ключевой буквы"
•
•         else:

```

```

•         if mode == "encrypt":
•             cipher_text_input = "Введите ключевую букву для шифра Виженера"
•         elif mode == "decrypt":
•             open_text_input = "Введите ключевую букву для шифра Виженера"
•     elif cipher_choose_input == "МАГМА(s_block)":
•         if mode == "encrypt":
•             cipher_text_input = s_block_encrypt(self.text_preparation(open_text_input), alphabet_sblock)
•         elif mode == "decrypt":
•             open_text_input = s_block_decrypt(cipher_text_input, alphabet_sblock)
•     elif cipher_choose_input == "Шифр Матричный":
•         input_matrix = list(map(int, matrix_input.split()))
•         matrix_input = [input_matrix[:3], input_matrix[3:6], input_matrix[6:]]
•         if matrix_input:
•             if matrix_check_parameters(matrix_input):
•                 if mode == "encrypt":
•                     cipher_text_input = matrix_encrypt(self.text_preparation(open_text_input), matrix_input,
alphabet)
•                 elif mode == "decrypt":
•                     open_text_input = matrix_decrypt(cipher_text_input, matrix_input, alphabet)
•             else:
•                 if mode == "encrypt":
•                     cipher_text_input = "Проверьте правильность ввода матрицы"
•                 elif mode == "decrypt":
•                     open_text_input = "Проверьте правильность ввода матрицы"
•             else:
•                 if mode == "encrypt":
•                     cipher_text_input = "Введите ключевую матрицу для шифра Матричный"
•                 elif mode == "decrypt":
•                     open_text_input = "Введите ключевую матрицу для шифра Матричный"
•     elif cipher_choose_input == "Шифр Плейфера":
•         keyword = self.keyword_edit.text()
•         if keyword:

```

```

•         if playfair_check_parameters(keyword, alphabet_playfair):
•             if mode == "encrypt":
•                 cipher_text_input = playfair_encrypt(self.text_preparation(open_text_input), keyword,
alphabet_playfair)
•             elif mode == "decrypt":
•                 open_text_input = playfair_decrypt(cipher_text_input, keyword, alphabet_playfair)
•         else:
•             if mode == "encrypt":
•                 cipher_text_input = "Проверьте правильность ключевого слова"
•             elif mode == "decrypt":
•                 open_text_input = "Проверьте правильность ключевого слова"
•         else:
•             if mode == "encrypt":
•                 cipher_text_input = "Введите ключевое слово для шифра Плейфэра"
•             elif mode == "decrypt":
•                 open_text_input = "Введите ключевое слово для шифра Плейфэра"
•         # elif cipher_choose_input == "Вертикальная Транспозиция":
•         #     keyword = self.vertical_transposition_keyword_edit.text()
•         #     if keyword:
•         #         matrix_input = [int(num) for num in keyword.split() if num.isdigit()]
•         #         if vertical_transposition_check_parameters(matrix_input, cipher_text_input):
•         #             if mode == "encrypt":
•         #                 cipher_text_input = vertical_transposition_encrypt(cipher_text_input, matrix_input)
•         #             elif mode == "decrypt":
•         #                 open_text_input = vertical_transposition_decrypt(cipher_text_input, matrix_input)
•         #         else:
•         #             if mode == "encrypt":
•         #                 cipher_text_input = "Проверьте правильность ввода ключа"
•         #             elif mode == "decrypt":
•         #                 open_text_input = "Проверьте правильность ввода ключа"
•         #     else:
•         #         if mode == "encrypt":

```

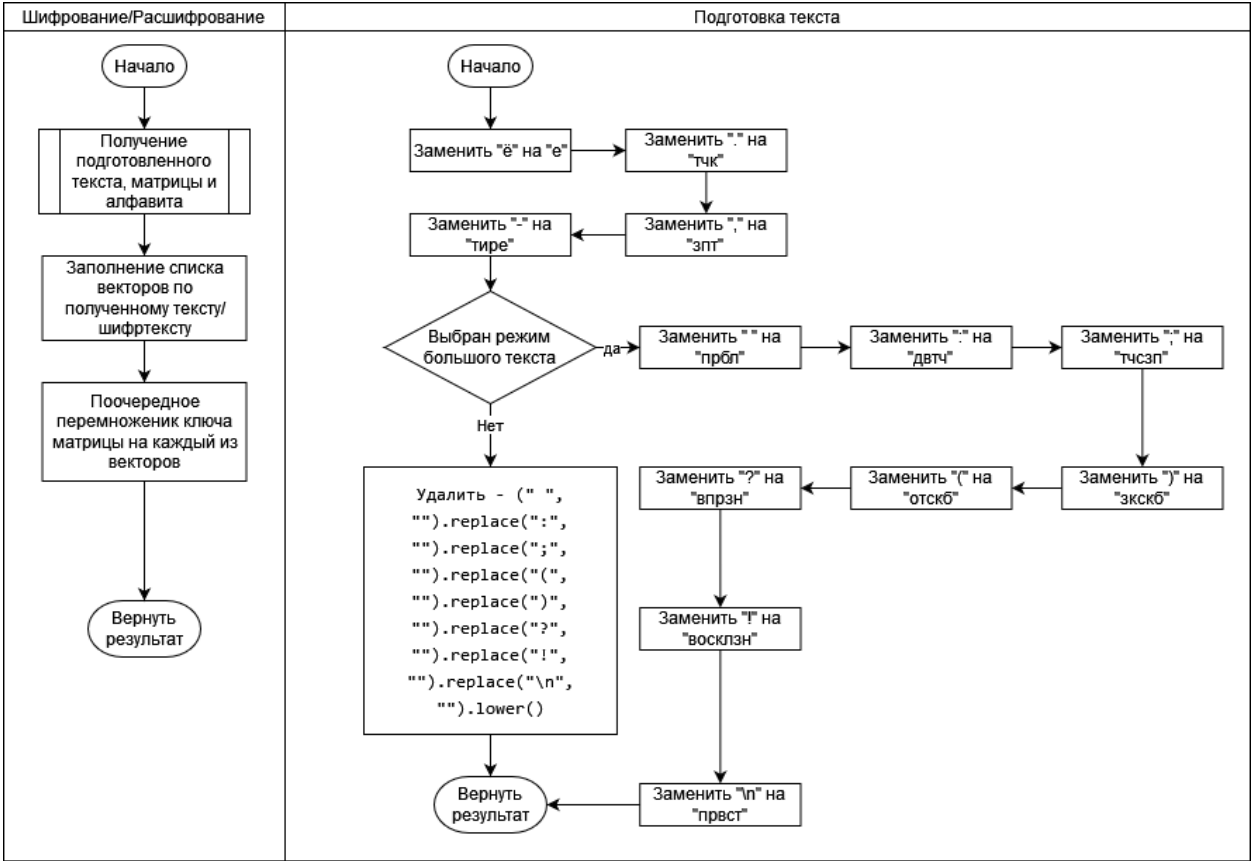
```
•         cipher_text_input = "Введите ключ и текст для шифра вертикальной транспозиции"
•         # elif mode == "decrypt":
•         #         open_text_input = "Введите ключ и текст для шифра вертикальной транспозиции"
•     else:
•         pass
•
•     # Обновление текста в виджетах
•     self.open_text_edit.setPlainText(open_text_input)
•     self.cipher_text_edit.setPlainText(cipher_text_input)
•
• if __name__ == '__main__':
•     app = QApplication(sys.argv)
•     ex = CipherApp()
•     ex.show()
•     sys.exit(app.exec_())
```

8. Матричный шифр

Принцип работы шифра можно разделить на несколько шагов:

1. Генерация ключа — создание квадратной матрицы, которая называется ключевой матрицей.
2. Преобразование текста в числовую форму — каждый символ текста преобразуется в числовое значение.
3. Умножение ключевой матрицы на векторы текста — ключевая матрица умножается на каждый вектор текста.
4. Дешифрование текста — для дешифрования текста выполняются обратные операции.

Блок-схема программы



Код программы с комментариями

```
def multiply_matrix(A, B):
    rowsA, colsA = len(A), len(A[0])
    rowsB, colsB = len(B), len(B[0])
    C = []
    if colsA != rowsB:
        return False
    for i in range(rowsA):
        C.append([])
    for k in range(colsB):
        for i in range(rowsA):
            t = 0
            for j in range(rowsB):
                t += A[i][j] * B[j][k]
            C[i].append(t)
    return C

def determinant(A):
    N = len(A)
    B = [row[:] for row in A]
    denom = 1
    exchanges = 0
    for i in range(N - 1):
        maxN = i
        maxValue = abs(B[i][i])
        for j in range(i + 1, N):
            value = abs(B[j][i])
            if value > maxValue:
                maxN = j
                maxValue = value
        if maxN > i:
            B[i], B[maxN] = B[maxN], B[i]
```

```

        exchanges += 1
    else:
        if maxValue == 0:
            return maxValue
        value1 = B[i][i]
        for j in range(i + 1, N):
            value2 = B[j][i]
            B[j][i] = 0
            for k in range(i + 1, N):
                B[j][k] = (B[j][k] * value1 - B[i][k] * value2) / denom
        denom = value1
    if exchanges % 2:
        return -B[N - 1][N - 1]
    else:
        return B[N - 1][N - 1]

def adjugate_matrix(A):
    N = len(A)
    adjA = []
    for i in range(N):
        adjA.append([])
        for j in range(N):
            B = []
            sign = 1 if (i + j) % 2 == 0 else -1
            for m in range(j):
                B.append([A[m][n] for n in range(i)] + [A[m][n] for n in range(i + 1, N)])
            for m in range(j + 1, N):
                B.append([A[m][n] for n in range(i)] + [A[m][n] for n in range(i + 1, N)])
            adjA[i].append(sign * determinant(B))
    return adjA

def inverse_matrix(A):

```



```

    det = determinant(A)
    if det == 0:
        return False
    N = len(A)
    B = adjugate_matrix(A)
    for i in range(N):
        for j in range(N):
            B[i][j] /= det
    return B

def matrix_check_parameters(matrix):
    print(matrix)
    if determinant(matrix) == 0:
        return False
    for row in matrix:
        for num in row:
            if not isinstance(num, (int, float)):
                return False
    if len(matrix) == 3 and len(matrix[2]) == 3:
        return True # True, если матрица 3x3
    return False # False, если матрица не соответствует требованиям

def matrix_encrypt(open_text, key_matrix, alphabet):
    encrypted_text = ""
    open_text_array = []
    if len(open_text) % len(key_matrix) != 0:
        open_text += "ϕ" * (3 - len(open_text) % len(key_matrix))
    for i in range(0, len(open_text), len(key_matrix)):
        vector = []
        for letter in open_text[i:i + len(key_matrix)]:
            vector.append([alphabet.index(letter)])
        open_text_array.append(vector)

```

```

    for vector in open_text_array:
        result_matrix = multiply_matrix(key_matrix, vector)
        if not result_matrix:
            break
        for el in result_matrix:
            encrypted_text += "".join(str(e).rjust(3, "0") for e in el)
    return encrypted_text

def matrix_decrypt(encrypted_text, key_matrix, alphabet):
    decrypted_text = ""
    inverse_matrix_ = inverse_matrix(key_matrix)
    encrypted_text_array = []
    for i in range(0, len(encrypted_text), 9):
        vector = []
        current_str = encrypted_text[i:i + 9]
        for j in range(len(key_matrix)):
            vector.append([int(current_str[j * 3:j * 3 + 3])])
        encrypted_text_array.append(vector)
    for vector in encrypted_text_array:
        if not inverse_matrix_:
            break
        result_matrix = multiply_matrix(inverse_matrix_, vector)
        if not result_matrix:
            break
        for el in result_matrix:
            index = int(round(el[0][0].real)) if isinstance(el[0][0], complex) else int(round(el[0][0]))
            if index >= 0:
                decrypted_text += alphabet[index % len(alphabet)]
            else:
                decrypted_text += alphabet[index % len(alphabet)]
    return decrypted_text

```

Тестирование

Шифры

Выберите шифр:

Шифр Матричный

Введите открытый текст (Расшифрованный):

Тот, кто ложится на два стула, падает на ребра.

Шифрованный текст:

142172252136097195142120236099111198149109227096243303014032042164174283032084073150156267020008036134084205095037149081039150163182325

Введите сдвиг для шифра Цезаря

Введите ключевое слово для шифра Белазо или Плейфера

Введите ключевую букву для шифра Виженера

153721691

Выберите режим:

Шифрование

Выполнить

☐ Расширенный текст

14217	22521	36971	95142
22023	69911	14981	49109
22796	24330	31432	42164
17428	33284	73450	15626
72083	61348	42059	53714
98139	15016	31823	25

Работа с текстом не менее 1000 знаков

Зашифрование

Шифры

Выберите шифр:

Шифр Матричный

Введите открытый текст(Расшифрованный):

Жизнь - это удивительное приключение, полное разнообразных событий и встреч. В каждом моменте мы находим что-то новое и уникальное. Стремление к росту и саморазвитию вдохновляет нас на поиск новых горизонтов. Важно помнить, что каждый шаг вперед приносит с собой уроки и опыт.

Разнообразие культур, языков и традиций делает наш мир удивительно богатым. Общение с людьми разных национальностей расширяет кругозор, позволяя нам понимать и уважать друг друга. Взаимное уважение и терпимость создают основу для гармоничного сосуществования.

Природа тоже играет важную роль в нашей жизни. Красота закатов, шум океана, пение птиц - все это напоминает нам о величии мира природы. Забота о окружающей среде становится неотъемлемой частью ответственного образа жизни.

Работа и творчество придают смысл нашим усилиям. Стремление к достижению целей мотивирует нас на новые начинания. Каждый проект, даже самый маленький, приносит удовлетворение и чувство выполненного долга.

Семья и друзья являются надежной опорой в нашей жизни. Обмен историями, веселые посиделки и поддержка в трудные моменты создают теплую атмосферу взаимопомощи и любви.

Шифрованный текст:

0670651151981623450541251161061581961280811811430581341331692490541251160631491580960481021440851570981242090981382351341232170910871491602613920681121551360971950981382351281212151181171971280811811040451210740270760871272110371191032111663420981382351381252331900792670980891440541251161311021990620311071551712801740992551181962430541251161250601630860391150420160581191372070541251161181242101240791651280811810920411171502353130950421181470562030805610809813823518014129112814225409514015913714423509504211806611611612808118108003711309813823514512825005807613819014633109312614711819624305412511615517128009807014909406915512808118108603911509813823513312322416417428309813823509610815305412511605313111409413022804106106814817221009813823503308508815815328605711311311024524909813823507610318314016525309504211812304615110104412010513117313311621109504211810512912914419327706503210811813823611609018109415623616318232509813823502108108411308216709813823512812121510111819717918936715015626705412511615521131409813823506109715610707709913210920512805312906002104205412511609204915205312614509813823513412321711509617911115319209813823514112623405412511609014822910413118005412511614117927209510114705412511613110219909804311920417735116318232510113923615217027907713313109815825211408917306711610905902207112808118108603911515818324116118029513609719509813823518714635211922326612506016308003711309813823514912924404401604408917921309813823504609010702608207114117225909504211816506323105412511610011616009813823511811924104206807407614716413523629909813823505809308905703903614123128811819624305412511609412511809406915512808118110704612209813823515612919518009628813110219910404512107402707617724636605412511607911310011709718713905012713413622107014516209813823509110921016117532618617938014117225908603911512015327009112616109614117414117225910104412005511414916215121415426333709504211810503912305412511612414622906808015620319737505412511613110219911304812407002001817406419000813823507010111807015415605412511614107018705203903611819624305412511603707807510709316908417314405412511607013713207006509

Введите сдвиг для шифра Цезаря

Введите ключевое слово для шифра Белазо или Плейфера

Введите ключевую букву для шифра Виженера

1 5 3 7 2 1 6 9 1

Выберите режим:

Шифрование

Выполнить

☒ Расширенный текст

Расшифрование

Шифры

Выберите шифр:Шифр Матричный

Введите открытый текст (Расшифрованный):

Жизнь - это удивительное приключение, полное разнообразных событий и встреч. В каждом моменте мы находим что-то новое и уникальное. Стремление к росту и саморазвитию вдохновляет нас на поиск новых горизонтов. Важно помнить, что каждый шаг вперед приносит с собой уроки и опыт.

Разнообразие культур, языков и традиций делает наш мир удивительно богатым. Общение с людьми разных национальностей расширяет кругозор, позволяя нам понимать и уважать друг друга. Взаимное уважение и терпимость создают основу для гармоничного сосуществования.

Природа тоже играет важную роль в нашей жизни. Красота закатов, шум океана, пение птиц - все это напоминает нам о величии мира природы. Забота о окружающей среде становится неотъемлемой частью ответственного образа жизни.

Работа и творчество придают смысл нашим усилиям. Стремление к достижению целей мотивирует нас на новые начинания. Каждый проект, даже самый маленький, приносит удовлетворение и чувство выполненного долга.

Семья и друзья являются надежной опорой в нашей жизни. Обмен историями, веселые посиделки и поддержка в трудные моменты создают теплую атмосферу взаимопонимания и любви.

Шифрованный текст:

04310610611428231505113512008220819008319115511023309509123223805113512005114416507215607811124312305615220305014623308921519706114813609126539404412215509121116905014623308320319707919318108319115507115509505311305604513221003710611213031330705014623308721321310924722307116712805113512008620317602907109510125426310524321908823224105113512008017913405311909502405404808320619405113512007619019608519313908319115505913109510527930906213709508418916805612809205014623311126726107419424208019715209222721806213709506015011408319115504710709505014623308821722902808213610626230107818913808823224105113512010125426306214613105513313908319115505311909505014623308520820511027226305014623307217414105113512005314011904613622603508606512429219005014623302707809309523926605113211409525725705014623303710918109523923906213709507817112006814909508119816208821190062137095099228114099255267032077095070184226074176161052154238094247310050146233015054093074173146050146233083203197062161188095254350096240249051135120101266305050146233031094157095206074087207183095203095051105024051135120047113134038113152050146233089215197076181160087216183050146233090219213051135120048144231077191170051135120093240261071170134051135120086203176065143095123303321094247310053152233098248263071176128050152254075177153064157106038083056083191155053119095125299221104263278091211169050146233094241325089242269080179134047107095050146233095230221032068032065182218050146233034094109026074077096243246062137095093210192051135120076184148050146233061163229036090072061163166096261300050146233055133085057120021105276285088232241051135120091211106055133139083191155074161095050146233123279165096228252086203176071155095053113056114303357051135120079184091075180168106223088092224203055151166050146233043124205089236311093249360096243246053119095063177264070175154075189168096243246068149095034103154129297186109295337062137095069150096051135120082208217032092152119302350051135120086203176080173095070044014120258144050146233067160109070181152051135120093210156057120021088232241051135120016043070071170152069171138051135120023086149055127208

Введите сдвиг для шифра Цезаря

Введите ключевое слово для шифра Белазо или Плейфера

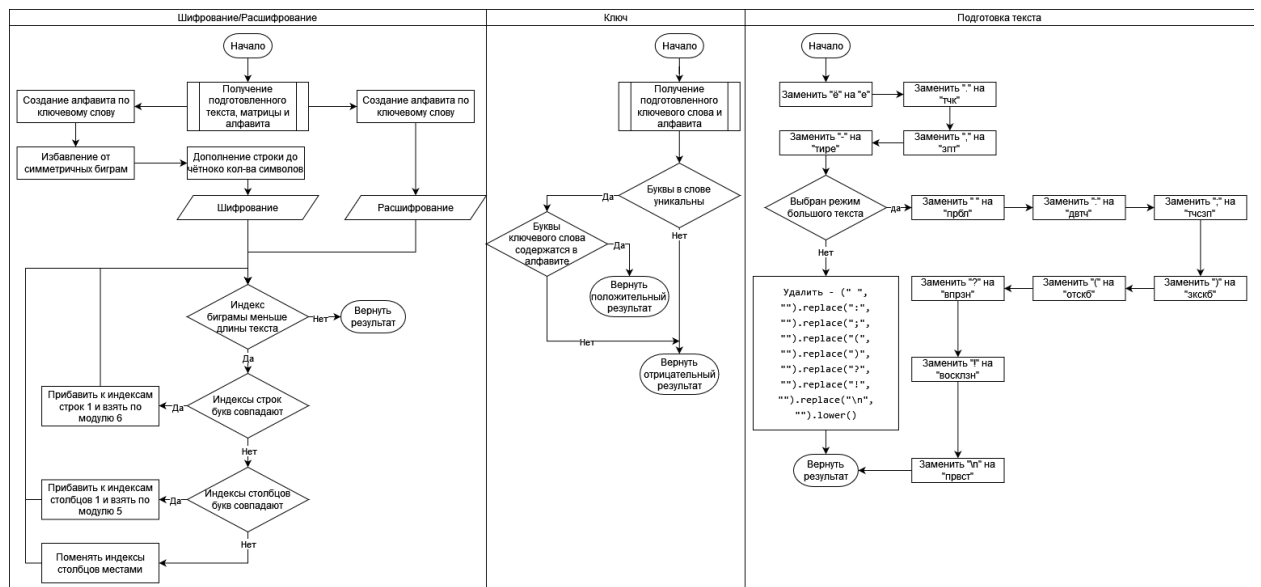
Введите ключевую букву для шифра Виженера

1 2 3 4 5 6 7 8 0

Выберите режим:Шифрование

Выполнить

☒ Расширенный текст



Код программы с комментариями

```
def get_alphabet_index(element, alphabet):
    for row in range(len(alphabet)):
        for col in range(len(alphabet[row])):
            if element == alphabet[row][col]:
                return [row, col]
    return []

def playfair_check_parameters(keyword, alphabet):
    if len(set(keyword)) != len(keyword):
        return False # Ложь, если буквы в слове не уникальны
    for letter in keyword:
        if letter not in alphabet:
            return False # Ложь, если буквы в ключевом слове не содержатся в алфавите
    return True # Истина, если всё ок

def playfair_encrypt(open_text, keyword, alphabet):
    encrypted_text = "" # Шифртекст
    # Формирование алфавита с учетом ключевого слова
    # print("keyword", keyword)
    # print("alphabet", alphabet)
    unprepared_alphabet = list(keyword) + [char for char in alphabet if char not in keyword]
    unprepared_alphabet = list(dict.fromkeys(unprepared_alphabet))
    # print(unprepared_alphabet, "unprepared_alphabet")

    while len(unprepared_alphabet) < 30:
        unprepared_alphabet.append('_')

    new_alphabet = [unprepared_alphabet[i:i+6] for i in range(0, len(unprepared_alphabet), 6)]

    # print(new_alphabet, "newAlphabet")
```

```

for i in range(0, len(open_text) - 1, 2): # Избавление от биграм из одинаковых букв
    if open_text[i] == open_text[i + 1] and not (open_text[i] == 'ф' and open_text[i + 1] == 'ф'): # Если биграма не "фф"
        open_text = open_text[:i + 1] + "ф" + open_text[i + 1:]
    elif open_text[i] == open_text[i + 1] and open_text[i] == 'ф' and open_text[i + 1] == 'ф': # Если биграма "фф"
        open_text = open_text[:i + 1] + "х" + open_text[i + 1:]

open_text += "ф" * (len(open_text) % 2) # Дополнение строки до чётного кол-ва букв

for i in range(0, len(open_text), 2): # Зашифрование
    first_index = get_alphabet_index(open_text[i], new_alphabet)
    second_index = get_alphabet_index(open_text[i + 1], new_alphabet)
    if first_index[0] == second_index[0]:
        first_index[1] = (first_index[1] + 1) % 6
        second_index[1] = (second_index[1] + 1) % 6
        encrypted_text += new_alphabet[first_index[0]][first_index[1]] + new_alphabet[second_index[0]][second_index[1]]
    elif first_index[1] == second_index[1]:
        first_index[0] = (first_index[0] + 1) % 5
        second_index[0] = (second_index[0] + 1) % 5
        encrypted_text += new_alphabet[first_index[0]][first_index[1]] + new_alphabet[second_index[0]][second_index[1]]
    else:
        first_index[1], second_index[1] = second_index[1], first_index[1]
        encrypted_text += new_alphabet[first_index[0]][first_index[1]] + new_alphabet[second_index[0]][second_index[1]]

return encrypted_text # Возврат шифртекста

def playfair_decrypt(encrypted_text, keyword, alphabet):
    decrypted_text = "" # Расшифрованный текст
    # Формирование алфавита с учетом ключевого слова
    unprepared_alphabet = list(keyword) + [char for char in alphabet if char not in keyword]
    unprepared_alphabet = list(dict.fromkeys(unprepared_alphabet))

```



```

while len(unprepared_alphabet) < 30:
    unprepared_alphabet.append('_')

new_alphabet = [unprepared_alphabet[i:i+6] for i in range(0, len(unprepared_alphabet), 6)]

for i in range(0, len(encrypted_text), 2): # Расшифрование
    first_index = get_alphabet_index(encrypted_text[i], new_alphabet)
    second_index = get_alphabet_index(encrypted_text[i + 1], new_alphabet)
    if first_index[0] == second_index[0]:
        first_index[1] = (first_index[1] - 1 + 6) % 6
        second_index[1] = (second_index[1] - 1 + 6) % 6
        decrypted_text += new_alphabet[first_index[0]][first_index[1]] + new_alphabet[second_index[0]][second_index[1]]
    elif first_index[1] == second_index[1]:
        first_index[0] = (first_index[0] - 1 + 5) % 5
        second_index[0] = (second_index[0] - 1 + 5) % 5
        decrypted_text += new_alphabet[first_index[0]][first_index[1]] + new_alphabet[second_index[0]][second_index[1]]
    else:
        first_index[1], second_index[1] = second_index[1], first_index[1]
        decrypted_text += new_alphabet[first_index[0]][first_index[1]] + new_alphabet[second_index[0]][second_index[1]]

decrypted_text = decrypted_text.replace("тчк", ".").replace("зпт", ",").replace("тире", "-").replace('прбл', ' ')
decrypted_text = decrypted_text.replace('двтч', ':').replace('тчсзп', ';').replace('отскб', '(').replace('зксб', ')').replace('впрзн', '?')
decrypted_text = decrypted_text.replace('восклзн', '!').replace('првст', '\n')
return decrypted_text # Возврат расшифрованного текста

```

Тестирование

Шифры

Выберите шифр:

Шифр Плейфера

Введите открытый текст(Расшифрованный):

Тот, кто ложится на два стула, падает на ребра.

Шифрованный текст:

пбсикслкапжптштюарнзлкшфгжксрбргкброирорсшу

Введите сдвиг для шифра Цезаря

народ

Введите ключевую букву для шифра Виженера

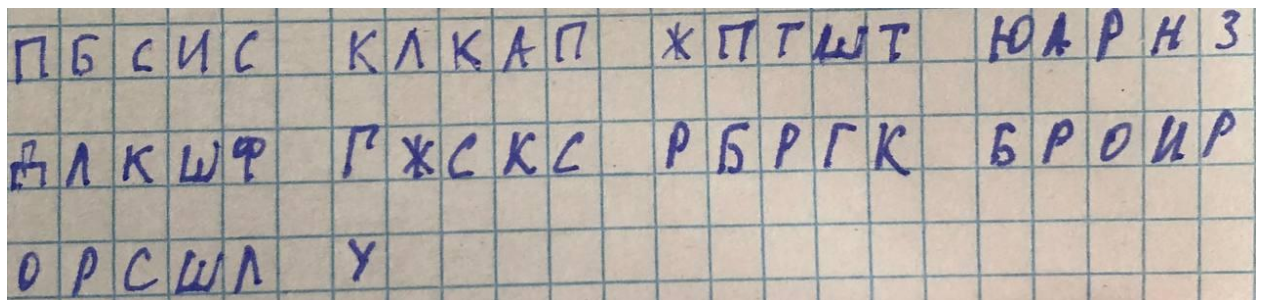
Введите ключевую матрицу для шифра Матричный

Выберите режим:

Шифрование

Выполнить

☐ Расширенный текст



Зашифрование

— □ ×

Шифр Плейфера

Введите открытый текст (Расшифрованный):

Жизнь - это удивительное приключение, полное разнообразных событий и встреч. В каждом моменте мы находим что-то новое и уникальное. Стремление к росту и саморазвитию вдохновляет нас на поиск новых горизонтов. Важно помнить, что каждый шаг вперед приносит с собой уроки и опыт.

Разнообразие культур, языков и традиций делает наш мир удивительно богатым. Общение с людьми разных национальностей расширяет кругозор, позволяя нам понимать и уважать друг друга. Взаимное уважение и терпимость создают основу для гармоничного сосуществования.

Природа тоже играет важную роль в нашей жизни. Красота закатов, шум океана, пение птиц - все это напоминает нам о величии мира природы. Забота о окружающей среде становится неотъемлемой частью ответственного образа жизни.

Работа и творчество придают смысл нашим усилиям. Стремление к достижению целей мотивирует нас на новые начинания. Каждый проект, даже самый маленький, приносит удовлетворение и чувство выполненного долга.

Семья и друзья являются надежной опорой в нашей жизни. Обмен историями, веселые посиделки и поддержка в трудные моменты создают теплую атмосферу взаимопонимания и любви.

Шифрованный текст:

[illegible]

Введите сдвиг для шифра Цезаря

народ

Введите ключевую букву для шифра Виженера

Введите ключевую матрицу для шифра Матричный

Выберите режим:

Шифрование

Выполнить

☒ Расширенный текст

Расшифрование

[illegible]