МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ
РОССИЙСКОЙ ФЕДЕРАЦИИ
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ УЧРЕЖДЕНИЕ
ВЫСШЕГО ОБРАЗОВАНИЯ
«МОСКОВСКИЙ ПОЛИТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»
ФАКУЛЬТЕТ ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ

ЛАБОРАТОРНЫЕ РАБОТЫ ПО ПРЕДМЕТУ

«Программирование криптографических алгоритмов»

Выполнил:

Барышников С.С. гр. 191-351

Преподаватель:

Бутакова Н.Г.

Москва 2021 г.

Содержание

Аннотация

Среда программирования: Visual Studio Code

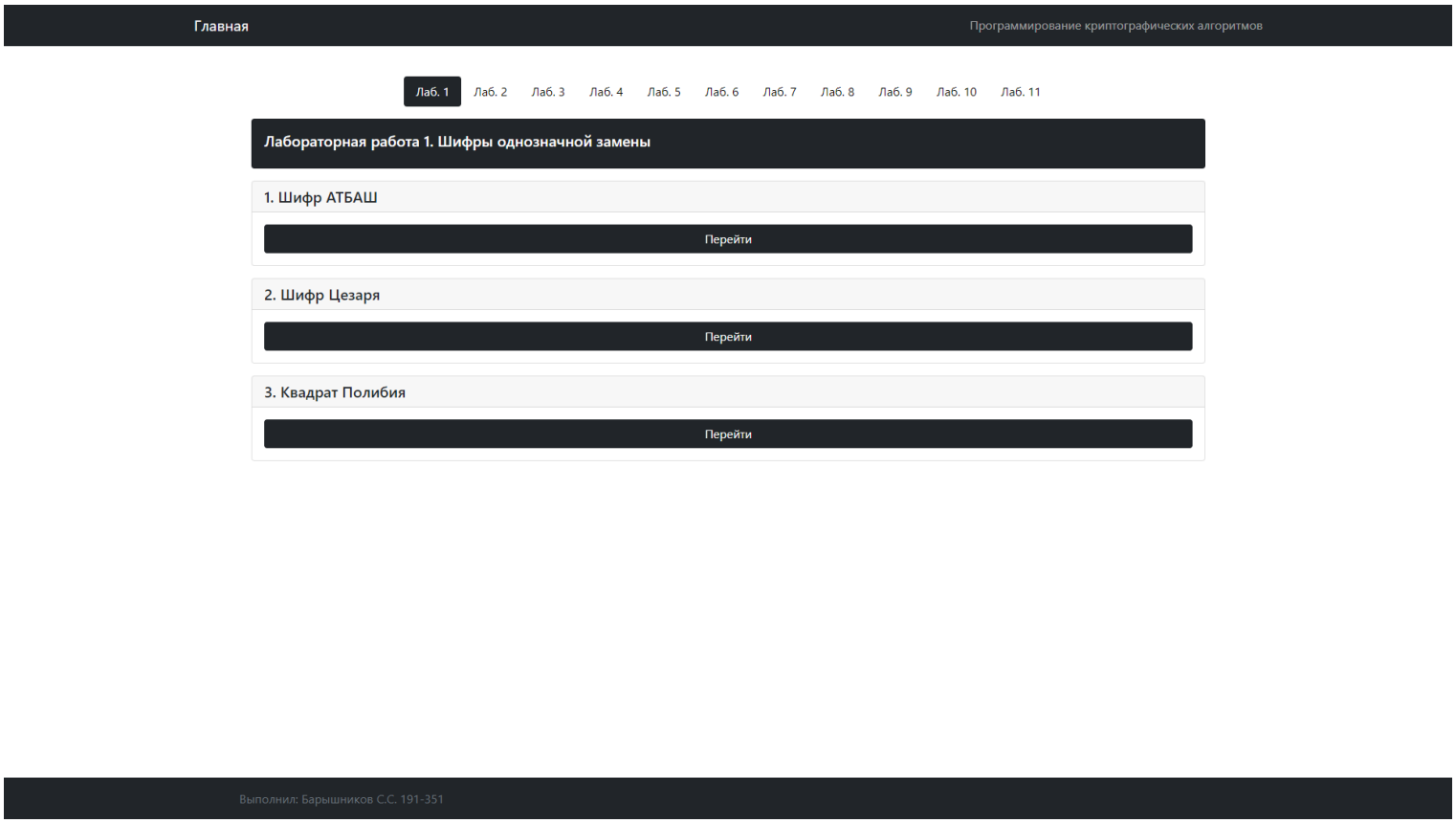
Язык программирования: Python 3

Процедуры для запуска программы: $ python3 <имя_файла>.py

Пословица-тест: Время, приливы и отливы не ждут человека.

Текст для проверки работы: Вот пример статьи на тысячу символов. Это достаточно маленький текст, оптимально подходящий для карточек товаров в интернет или магазинах или для небольших информационных публикаций. В таком тексте редко бывает более двух или трёх абзацев и обычно один подзаголовок. Но можно и без него. На тысячу символов рекомендовано использовать один или два ключа и одну картину. Текст на тысячу символов это сколько примерно слов? Статистика показывает, что тысяча включает в себя стопятьдесят или двести слов средней величины. Но, если злоупотреблять предлогами, союзами и другими частями речи на один или два символа, то количество слов неизменно возрастает. В копирайтерской деятельности принято считать тысячи с пробелами или без. Учет пробелов увеличивает объем текста примерно на сто или двести символов именно столько раз мы разделяем слова свободным пространством. Считать пробелы заказчики не любят, так как это пустое место. Однако некоторые фирмы и биржи видят справедливым ставить стоимость за тысячу символов с пробелами, считая последние важным элементом качественного восприятия. Согласитесь, читать слитный текст без единого пропуска, никто не будет. Но большинству нужна цена за тысячу знаков без пробелов.

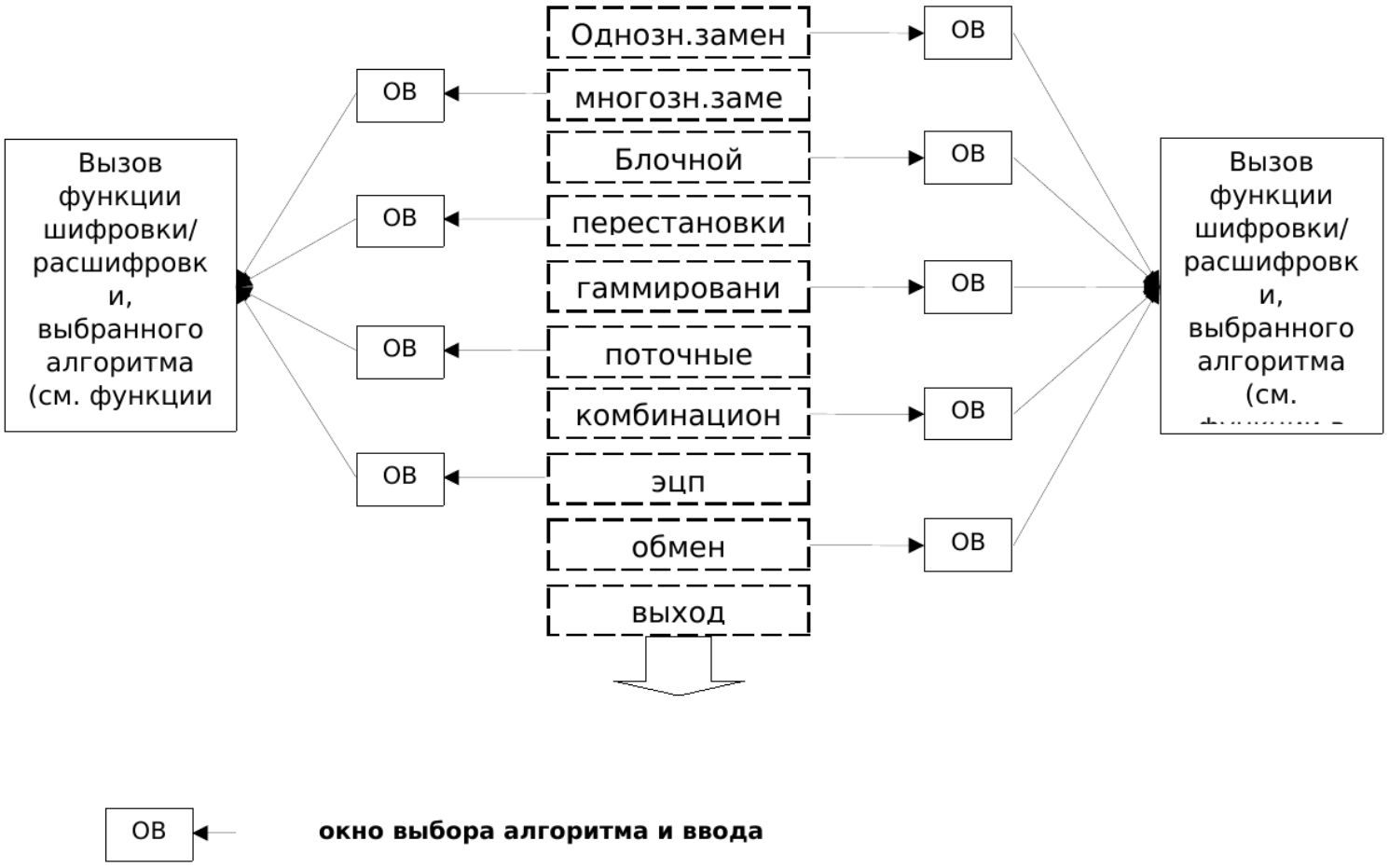
Интерфейс:



Интерфейс реализован с помощью фреймворка **Flask**. Программа имеет меню и разделение на страницы по блокам учебного курса.

Интерфейс можно протестировать онлайн [https://crypt.lsenichl.ru/.](https://crypt.lsenichl.ru/)

Блок-схема:



Код:

```
from flask import Flask, render_template
app = Flask(__name__)
application = app

@app.route('/')
def index():
return render_template('index.html')

from lab01 import bp as lab01_bp
app.register_blueprint(lab01_bp)

“““...“““

from lab11 import bp as lab11_bp
app.register_blueprint(lab11_bp)
```

Постоянный модуль

Код модуля base.py используемый для предотвращения дублирования кода, используется во всех последующих программах:

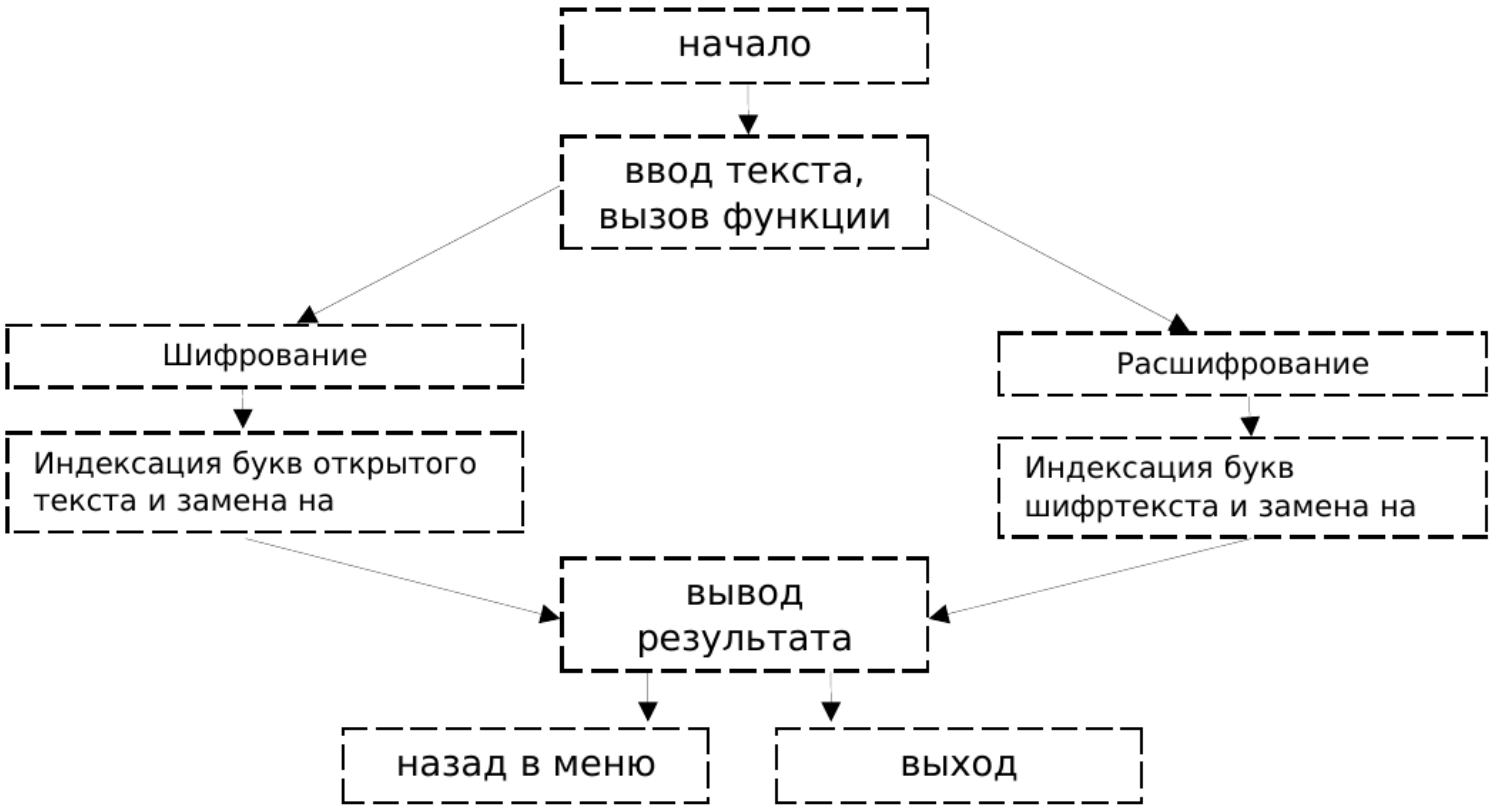
```
import re

alphabet = "абвгдеёжзийклмнопрстуфхцчшщъыьэюя"

dict = {'.': 'тчк', ',': 'зпт'}

def replace_all_to(input_text, dict):
    input_text = input_text.replace(' ', '')
    for i, j in dict.items():
        input_text = input_text.replace(i, j)
    return input_text

def replace_all_from(input_text, dict):
    for i, j in dict.items():
        input_text = input_text.replace(j, i)
    return input_text

def file_to_string(name):
    with open(name) as f:
        input_short_text = " ".join([l.rstrip() for l in f]) + ' '
    return input_short_text.lower()

def input_for_cipher_short():
    return replace_all_to(file_to_string('short.txt'), dict)

def input_for_cipher_long():
    return replace_all_to(file_to_string('long.txt'), dict)

def output_from_decrypted(decrypted_text):
    return replace_all_from(decrypted_text, dict)
```

Блок А: ШИФРЫ ОДНОЗНАЧНОЙ ЗАМЕНЫ

1. Шифр простой замены АТБАШ

Атбаш — простой шифр подстановки для алфавитного письма. Правило шифрования состоит в замене **i**-й буквы алфавита буквой с номером **n-i+1**, где **n** — число букв в алфавите.

Блок-схема:



Код программы:

```
# импорт компонентов, необходимых для работы программы
from base import alphabet, input_for_cipher_short,
input_for_cipher_long, output_from_decrypted

# функция шифрования/расшифрования
def atbash(input):
    return input.translate(str.maketrans(
        alphabet + alphabet.upper(), alphabet[::-1] + alphabet.upper()
[::-1]))

#вывод результатов работы программы
print(f'''
ШИФР АТБАШ:
КОРОТКИЙ ТЕКСТ:
Зашифрованный текст:
{atbash(input_for_cipher_short())}

Расшифрованный текст:
{output_from_decrypted(atbash(atbash(input_for_cipher_short())))}
```

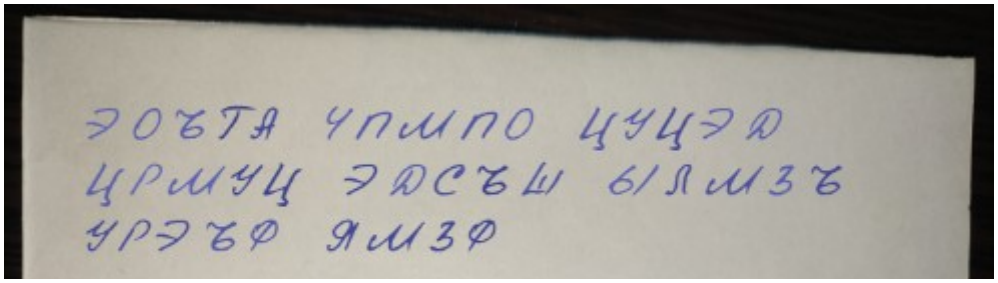
```
ДЛИННЫЙ ТЕКСТ:
Зашифрованный текст:
{atbash(input_for_cipher_long())}

Расшифрованный текст:
{output_from_decrypted(atbash(atbash(input_for_cipher_long())))}
''')
```

Тестирование:

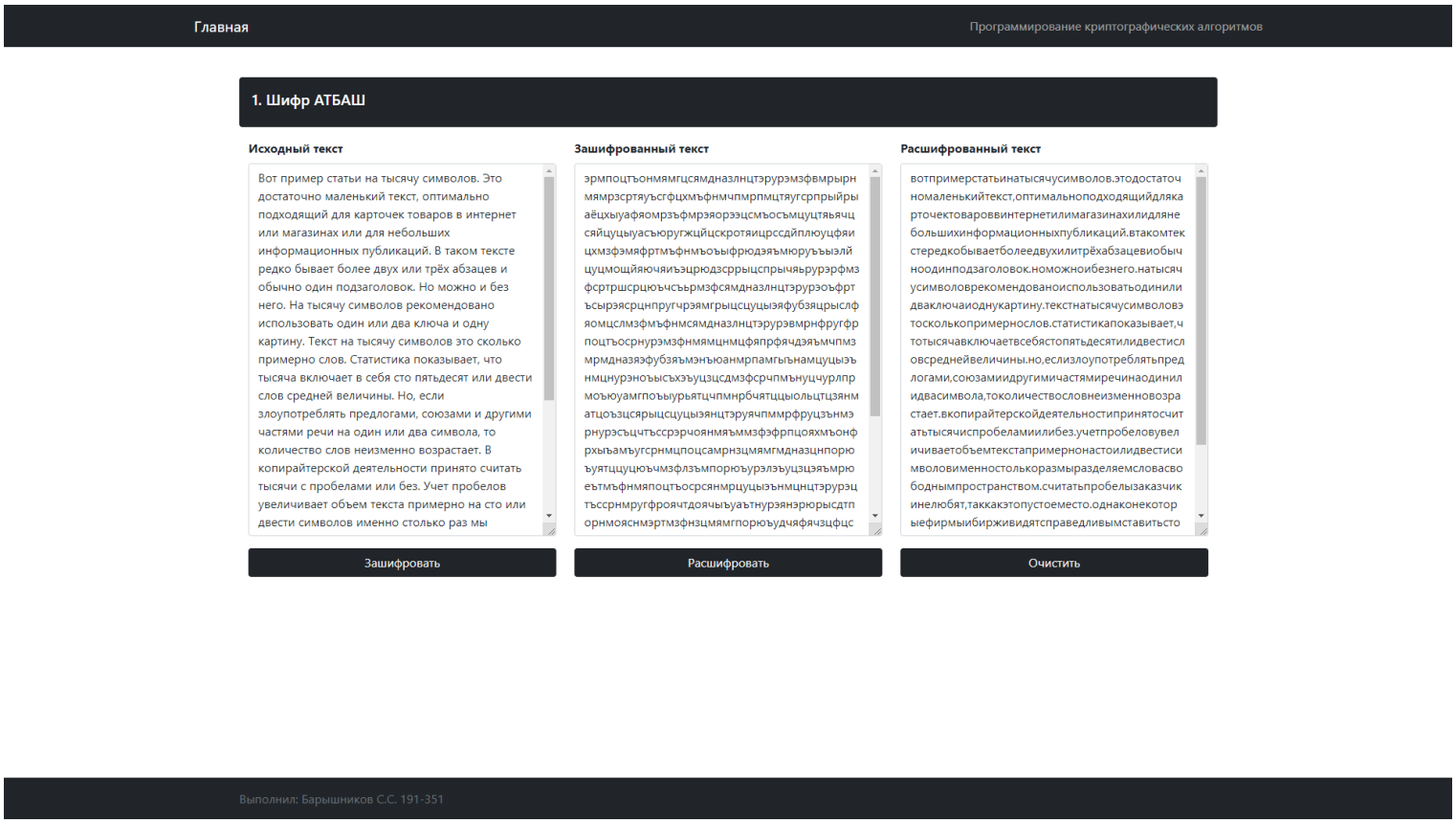
```
/bin/python3
/root/mospolytech-education-crypt-dev-2021-1/lab01_1_atbash.py
ШИФР АТБАШ:
КОРОТКИЙ ТЕКСТ:
Зашифрованный текст:
эоътачпмпоцуцэдцрмуцэдсъшылмзъурэъфямзф

Расшифрованный текст:
время,приливыиотливынеждутчеловека.

ДЛИННЫЙ ТЕКСТ:
Зашифрованный текст:
эрмпоцтъонмямгцсямдназлнцтэрурэмзфвмрырнмямрзсртяуъсгфцхмъфнмчпмрпмцтяуг
српрыйрыаёцхыуафяомрзъфмрэяорээцсмъосъмцуцтяьячцсяйцуцыуасъюругжцйцскротя
ицрссдйплюуцфяицхмзфэмяфртмъфнмъоъыфрюдэяъмюруъъыэлйцуцмощйяючяиъэцрюдзс
ррыцспрычяьрурэрфмзфсртршсрцюъчсъърмзфсямдназлнцтэрурэоъфртъсырэясрцнпру
гчрэямгрыцсцуцыэяфубзяцрыслфяомцслмзфмъфнмсямдназлнцтэрурэвмрнфругфрпоцт
ъосрнурэмзфнмямцнмцфяпрфячдэяъмчпмзмрмдназяэфубзяъмэнъюанмрпамгыънамцуцы
эънмцнурэноъысъхэъуцзцсдмзфсрчпмънуцчурлпрмоъюуамгпоъыурьятцчпмнрбчятццы
ольцтцзянматцоъзцсярыцсцуцыэянцтэруячпммрфруцзънмэрнурэсъцчтъссрэрчоянмя
ъммзфэфрпцояхмъонфрхыъамъугсрнмцпоцсамрнзцмямгмдназцнпорюъуятццуцюъчмзфл
зъмпорюъурэлэъуцзцэяъмрюеътмъфнмяпоцтъосрсянмрцуцыэънмцнцтэрурэцтъссрнмр
угфроячтдоячыъуаътнурэянэрюрысдтпорнмояснмэртмзфнзцмямгпорюъудчяфячзцфцс
ъубюамчпммяффяфвмрплнмрътънмрмзфрысяфрсъфрмродъкцотддюцошцэцыамнпояэъыуц
эдтнмяэцмгнмрцтрнмгчямдназлнцтэрурэнпорюъуятцчпмнзцмяапрнуъысцъэяшсдтвуъ
тъсмртфязънмэъссрьрэрнпоцамцамзфнрьуянцмънгчпмзцмямгнуцмсдхмъфнмюъчъыцср
ърпорплнфячпмсцфмрсъюлыъммзфсрюругжцснмэлслшсяиъсячямдназлчсяфрэюъчпорюъ
урэмзф

Расшифрованный текст:
вотпримерстатьинатысячусимволов.этодостаточномаленькийтекст,оптимальнопо
дходящийдлякарточектовароввинтернетилимагазинахилидлянебольшихинформацио
нныхпубликаций.втакомтекстередкобываетболеедвухилитрёхабзацевиобычноодинп
одзаголовок.номожноибезнего.натысячусимволоврекомендованоиспользоватьоди
нилидваключаиоднукартину.текстнатысячусимволовэтосколькопримернослов.ста
тистикапоказывает,чтотысячавключаетвсебястопятьдесятилидвестисловсредней
величины.но,еслизлоупотреблятьпредлогами,союзамиидругимичастямиречинаоди
нилидвасимвола,токоличествословнеизменновозрастает.вкопирайтерскойдеятел
ьностипринятосчитатьтысячиспробеламиилибез.учетпробеловувеличиваетобъемт
екстапримернонастоилидвестисимволовименностолькоразмыразделяемсловасвобо
днымпространством.считатьпробелызаказчикинелюбят,таккакэтопустоеместо.од
наконекоторыефирмыибиржividятсправедливымставитьстоимостьзатысячусимволо
```

**вспробелами,считаяпоследниеважнымэлементомкачественноговосприятия.соглас
итесь,читатьслитныйтекстбезединогопропуска,никтонебудет.нобольшинствунуж
наценазатысячузнаковбезпробелов.**

Карточка:



Интерфейс:

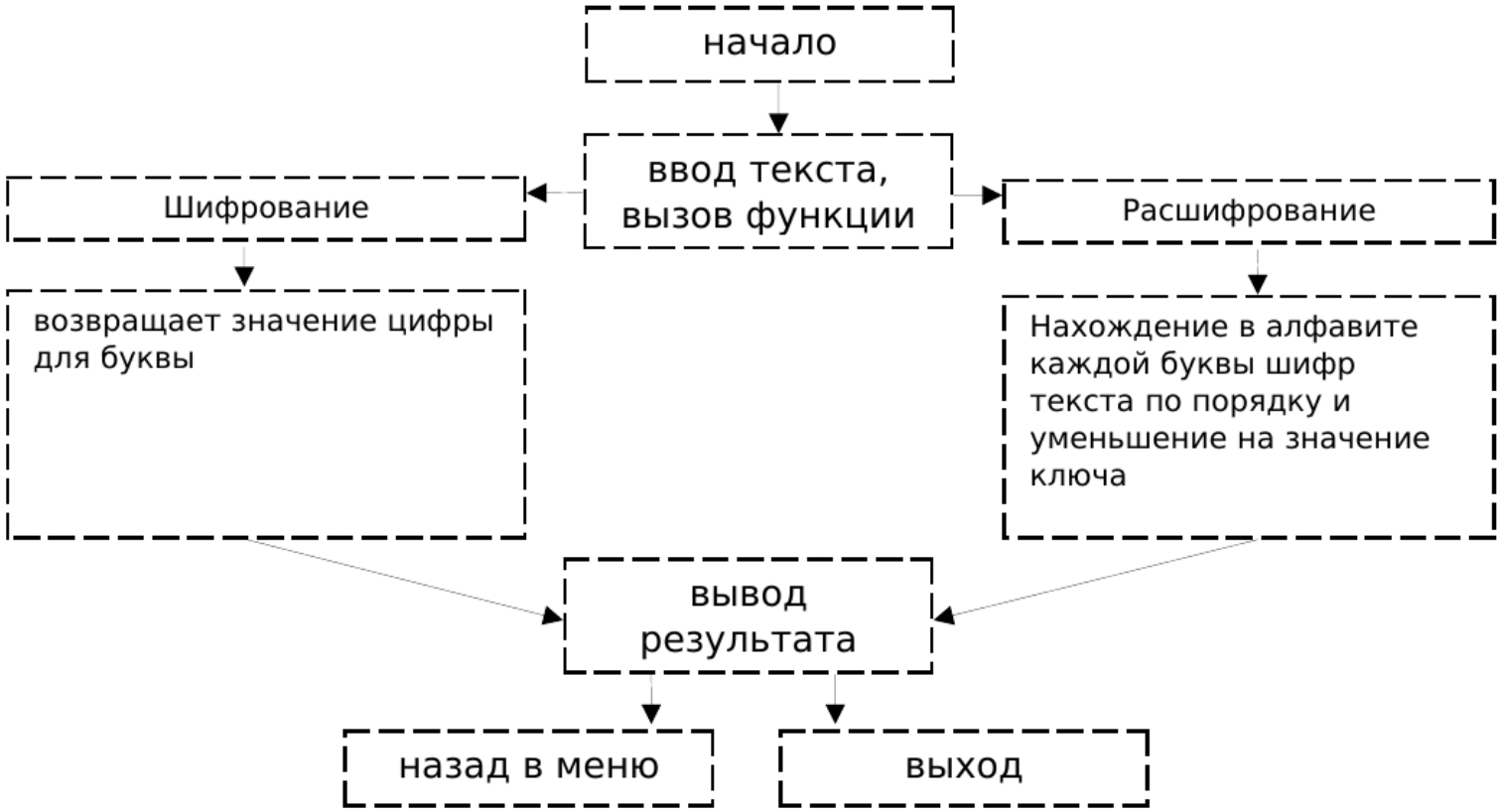


2. ШИФР ЦЕЗАРЯ

Шифр Цезаря, также известный как шифр сдвига, код Цезаря или сдвиг Цезаря — один из самых простых и наиболее широко известных методов шифрования.

Шифр Цезаря — это вид шифра подстановки, в котором каждый символ в открытом тексте заменяется символом, находящимся на некотором постоянном числе позиций левее или правее него в алфавите.

Блок-схема:



Код программы:

```
# импорт компонентов, необходимых для работы программы
from base import alphabet, input_for_cipher_short,
input_for_cipher_long, output_from_decrypted
import random

# установка ключа
key = 5

# функция шифрования
def caesar_encode(input, step):
    return input.translate(
        str.maketrans(alphabet, alphabet[step:] + alphabet[:step]))

# функция расшифрования
def caesar_decode(input, step):
    return input.translate(
        str.maketrans(alphabet[step:] + alphabet[:step], alphabet))
```

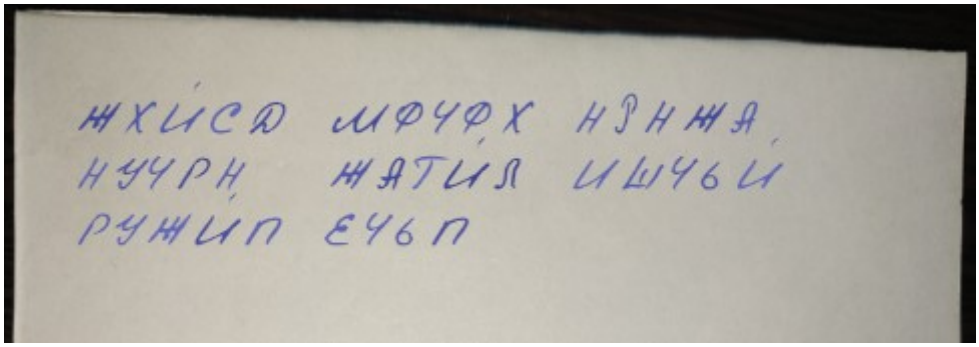
```
#вывод результатов работы программы
print(f'''
ШИФР ЦЕЗАРЯ:
Ключ: {key}
КОРОТКИЙ ТЕКСТ:
Зашифрованный текст:
{caesar_encode(input_for_cipher_short(), key)}

Расшифрованный текст:
{output_from_decrypted(caesar_decode(caesar_encode(
    input_for_cipher_short(), key), key))}

ДЛИННЫЙ ТЕКСТ:
Зашифрованный текст:
{caesar_encode(input_for_cipher_long(), key)}

Расшифрованный текст:
{output_from_decrypted(caesar_decode(caesar_encode(
    input_for_cipher_long(), key), key))}
''')
```

Тестирование:

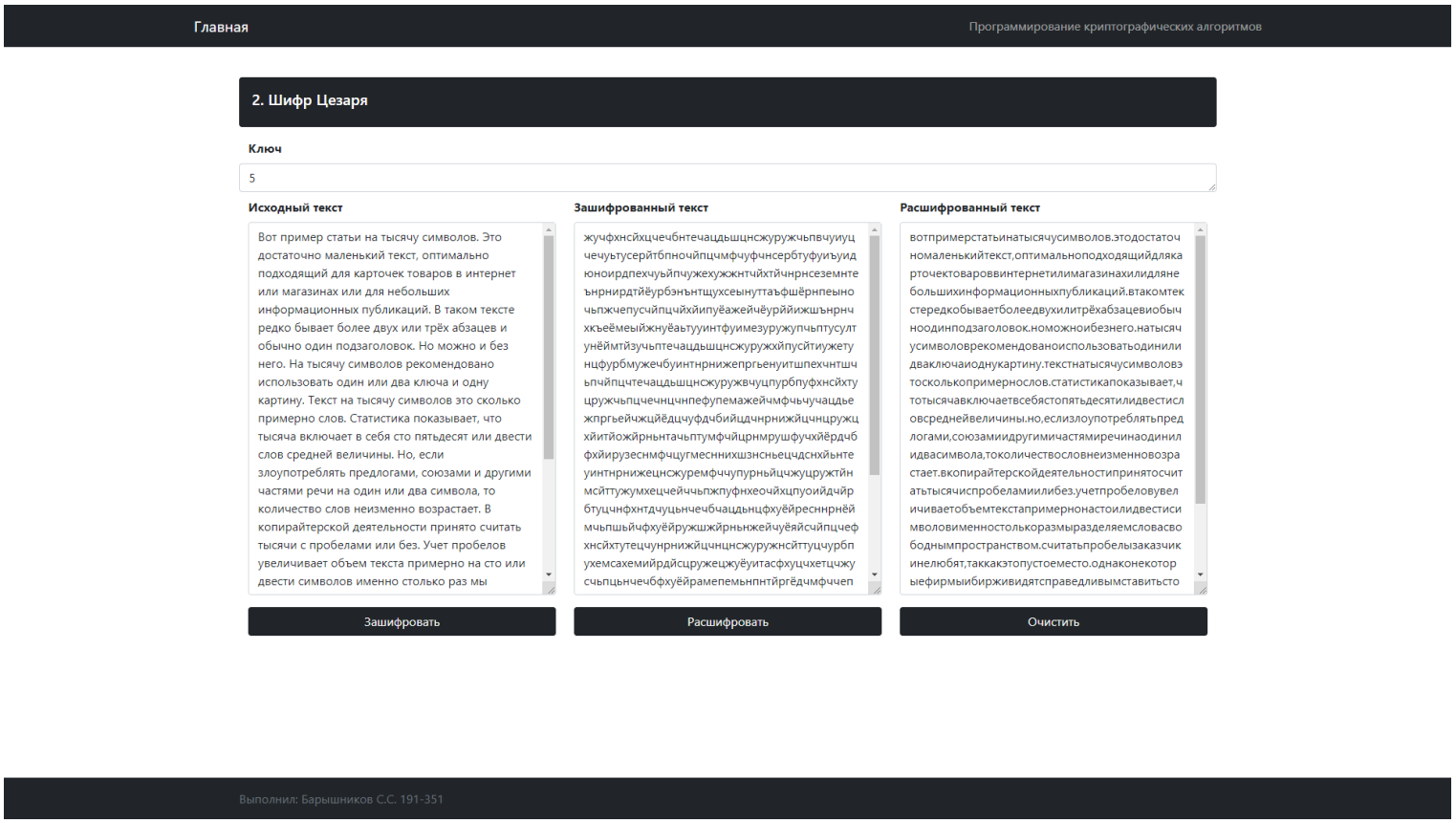
```
/bin/python3
/root/mospolytech-education-crypt-dev-2021-1/lab01_2_caesar.py

ШИФР ЦЕЗАРЯ:
Ключ: 5
КОРОТКИЙ ТЕКСТ:
Зашифрованный текст:
жхйсдмфчфхнрнжанучрнжатйлишчьйружйпечьп

Расшифрованный текст:
время,приливыиотливынеждутчеловека.

ДЛИННЫЙ ТЕКСТ:
Зашифрованный текст:
жучфхнсйхцчечбнтечацдьшцнсжуружчьпвчуиуцчечуьтусерйтбпночйпцчмфчуфчнсерб
туфуиъуидюноирдпехчуьйпчужехужжнтчйхтйчнрнсеземнтеънрнирдтйёурбэнънтщухсе
ынуттаъфшёрнпеыночьпжчепусчйпцчйхйипуёажейчёурййижшънрнчхкъеёмеыйжнуёаьту
уинтфуимезуружупчьптусултунёймтйзучьптечацдьшцнсжуружхйпусйтиужетунцфурбм
ужечбуинтнрнижепргьенуитшпехчнтшчьпчйпцчтечацдьшцнсжуружвчуцпурбпуфхнсйх
туцружчьпцчечнцчнпефупемажейчмфчьчучацдьежпргьейчжцйёдцчуфдчбийцдчнрнижйц
чнцружцхйитйожйрньнтачьптумфчйцрнмрушфучхйёрдчбфхйирузеснмфчцугмеснних шзн
сньецчдснхйьнтеуинтнрнижецнсжуремфччупурньйцчжуцружтйнмсйттужумхецчейччь
пжпуфнхеочйхцпуоийдчйрбтуцчнфхнтдчуцьнчечбчацдьнцфхуёйреснирнёймчьпшьйчфх
уёйружшжйрньнжейчуёяйсчйпцчефхнсйхтутецчунрнижйцчнцнсжуружнсйттуццчурбпухе
мсахемийрдйсцружецжуёуитасфхуцчхетцчжусчьпцьнчечбфхуёйрамепемьнпнтйргёдчм
фччеппепвчуфшцчуйсйцчучьпуитепутйпучухайщнхсанёнхлнжнидчцфхежйирнжасцчежн
чбцчунсуцчбмечацдьшцнсжуружцфхуёйреснмфчцьнчедфуцрйитнйжелтасврйсйтчуспеь
йцчжйттузужуцфхндчндчьпцузрецнчйцбмфчьнчечбцрнчтаочйпцчёймйинтузуфхуфшцпе
мфчтнпчутйёшийччьптуёурбэнтцчжштшлтеыйтемечацдьшмтепужёймфхуёйружчьп
```

```
Расшифрованный текст:
вотпримерстатьинатысячусимволов.этодостаточномаленькийтекст,оптимальнопо
дходящийдлякарточектовароввинтернетилимагазинахилидлянебольшихинформацио
нныхпубликаций.втакомтекстередкобываетболеедвухилитрёхабзацевиобычноодинп
одзаголовок.номожноибезнего.натысячусимволоврекомендованоиспользоватьоди
нилидваключаиоднукартину.текстнатысячусимволовэтосколькопримернослов.ста
тистикапоказывает,чтотысячавключаетвсебястопятьдесятилидвестисловсредней
величины.но,еслизлоупотреблятьпредлогами,союзамиидругимичастямиречинаоди
нилидвасимвола,токоличествословнеизменновозрастает.вкопирайтерскойдеятел
ьноститипринятосчитатьтысячиспробеламиилибез.учетпробеловувеличиваетобъемт
екстапримернонастоилидвестисимволовименностолькоразмыразделяемсловасвобо
днымпространством.считатьпробелызаказчикинелюбят,таккакэтопустоеместо.од
наконекоторыефирмыибиржividятсправедливымставитьстоимостьзатысячусимволо
вспробелами,считаяпоследниеважнымэлементомкачественноговосприятия.соглас
итесь,читатьслитныйтекстбезединогопропуска,никтонебудет.нобольшинствунуж
наценазатысячузнаковбезпробелов.
```

Карточка:



Интерфейс:

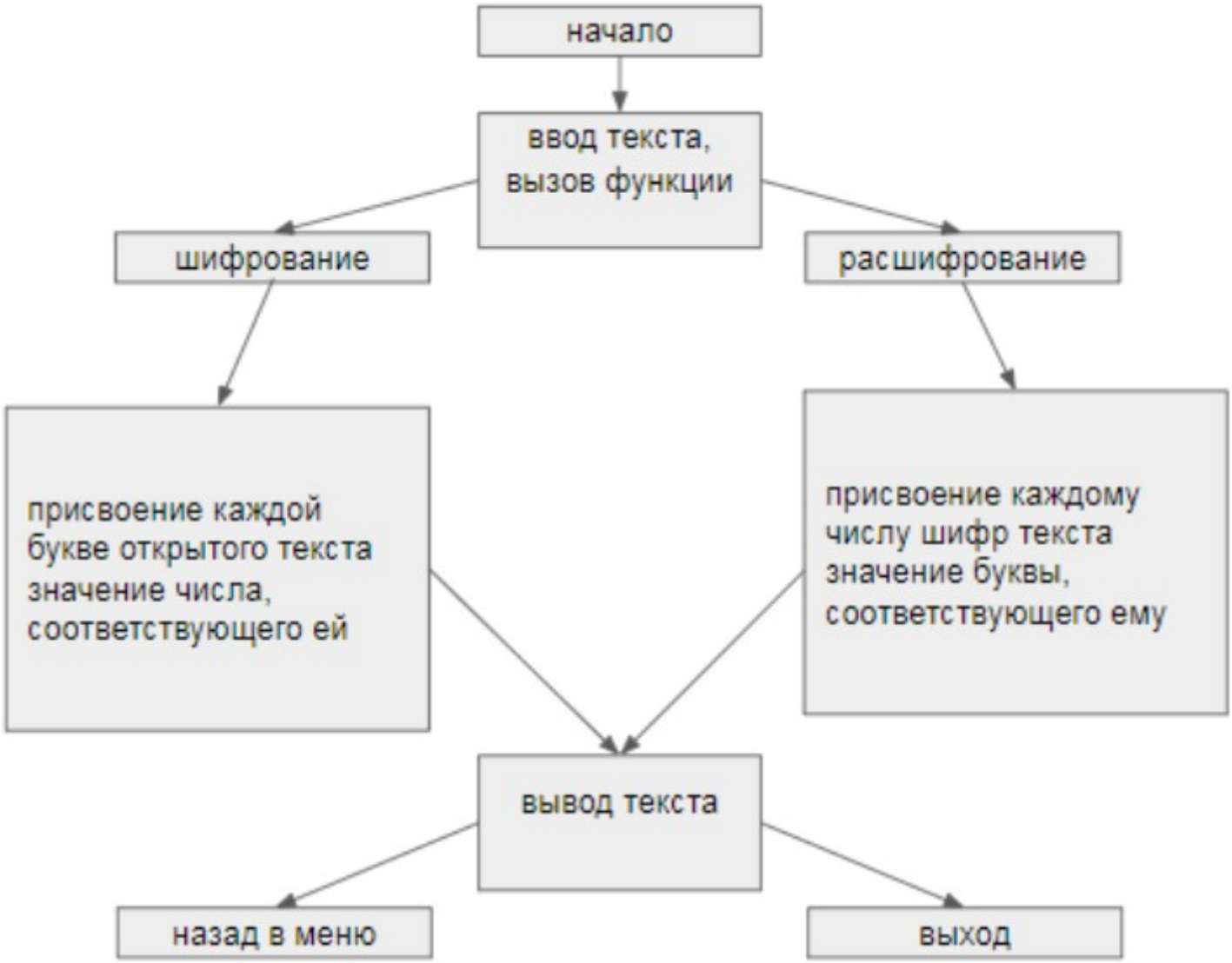


3. Квадрат Полибия

Квадрат Полибия – метод шифрования текстовых данных с помощью замены символов, впервые предложен греческим историком и полководцем Полибием.

К каждому языку отдельно составляется таблица шифрования с одинаковым (не обязательно) количеством пронумерованных строк и столбцов, параметры которой зависят от его мощности (количества букв в алфавите). Берутся два целых числа, произведение которых ближе всего к количеству букв в языке — получаем нужное число строк и столбцов. Затем вписываем в таблицу все буквы алфавита подряд — по одной на каждую клетку. При нехватке клеток можно вписать в одну две буквы (редко употребляющиеся или схожие по употреблению).

Блок-схема:



Код программы:

```
# импорт компонентов, необходимых для работы программы
from base import alphabet, input_for_cipher_short,
input_for_cipher_long, output_from_decrypted

# объявление алфавита
```

```
hard_dictionary = {"а": "11", "б": "12", "в": "13",
                   "г": "14", "д": "15", "е": "16", "ё": "21",
                   "ж": "22", "з": "23", "и": "24", "й": "25",
                   "к": "26", "л": "31", "м": "32", "н": "33",
                   "о": "34", "п": "35", "р": "36", "с": "41",
                   "т": "42", "у": "43", "ф": "44", "х": "45",
                   "ц": "46", "ч": "51", "ш": "52", "щ": "53",
                   "ъ": "54", "ы": "55", "ь": "56", "э": "61",
                   "ю": "62", "я": "63"}

# функция шифрования
def square_encode(input):
    new_txt = ""
    for x in input:
        if x in hard_dictionary:
            new_txt += hard_dictionary.get(x)
        else:
            new_txt += (x + x)
    return new_txt

# функция расшифрования
def square_decode(input):
    new_txt = ""
    list_fraze = []
    step = 2
    for i in range(0, len(input), 2):
        list_fraze.append(input[i:step])
        step += 2
    key_hard_dictionary_list = list(hard_dictionary.keys())
    val_hard_dictionary_list = list(hard_dictionary.values())

    for x in list_fraze:
        if x in val_hard_dictionary_list:
            i = val_hard_dictionary_list.index(x)
            new_txt += key_hard_dictionary_list[i]
        else:
            new_txt += x[0:1]
    return new_txt

# вывод результатов работы программы
print(f'''
КВАДРАТ ПОЛИБИЯ:
Ключ: {hard_dictionary}
КОРОТКИЙ ТЕКСТ:
Зашифрованный текст:
{square_encode(input_for_cipher_short())}

Расшифрованный текст:
{output_from_decrypted(square_decode(square_encode(
    input_for_cipher_short())))}

ДЛИННЫЙ ТЕКСТ:
Зашифрованный текст:
```

```
{square_encode(input_for_cipher_long())}

Расшифрованный текст:
{output_from_decrypted(square_decode(square_encode(
    input_for_cipher_long())))}
''')
```

Тестирование:

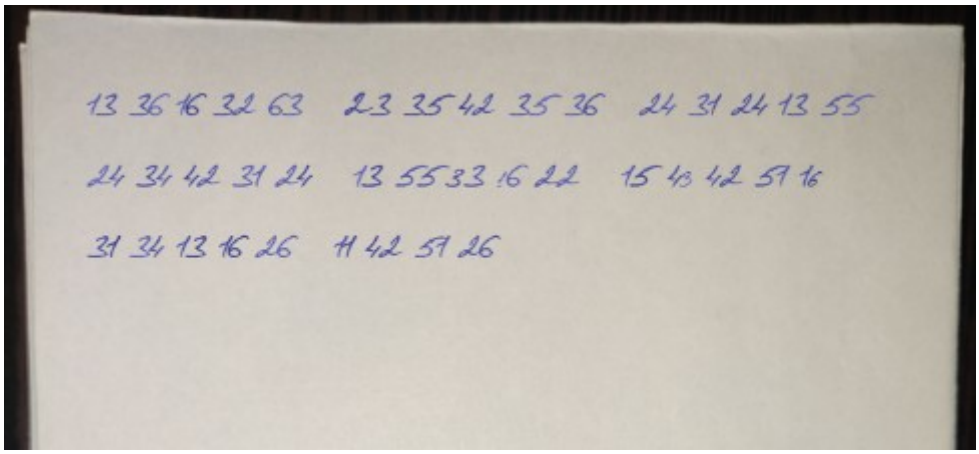
```
/bin/python3
/root/mospolytech-education-crypt-dev-2021-1/lab01_3_square.py

КВАДРАТ ПОЛИБИЯ:
Ключ: {'а': '11', 'б': '12', 'в': '13', 'г': '14', 'д': '15', 'е': '16',
'ё': '21', 'ж': '22', 'з': '23', 'и': '24', 'й': '25', 'к': '26', 'л':
'31', 'м': '32', 'н': '33', 'о': '34', 'п': '35', 'р': '36', 'с': '41',
'т': '42', 'у': '43', 'ф': '44', 'х': '45', 'ц': '46', 'ч': '51', 'ш':
'52', 'щ': '53', 'ъ': '54', 'ы': '55', 'ь': '56', 'э': '61', 'ю': '62',
'я': '63'}
КОРОТКИЙ ТЕКСТ:
Зашифрованный текст:
133616326323354235362431241355243442312413553316221543425116313413162611
425126

Расшифрованный текст:
время,приливыиотливынеждутчеловека.

ДЛИННЫЙ ТЕКСТ:
Зашифрованный текст:
133442353624321636414211425624331142554163514341243213343134134251266142
341534414211423451333432113116335626242542162641422335423435422432113156
333435341545341563532425153163261136423451162642341311363413132433421636
331642243124321114112324331145243124153163331612343156522445243344343632
114624343333554535431231242611462425425126134211263432421626414216361615
263412551311164212343116161513434524312442362145111223114616132434125551
333434152433353415231114343134133426425126333432342233342412162333161434
425126331142554163514341243213343134133616263432163315341311333424413534
315623341311425634152433243124151311263162511124341533432611364224334342
512642162641423311425541635143412432133431341361423441263431562634353624
321636333441313413425126414211422441422426113534261123551311164223354251
423442554163511113263162511116421341161263414234356342561516416342243124
151316414224413134134136161533162513163124512433554251263334233542164131
242331344335344236161231634256353616153134141132242335424134622311322424
153643142432245111414263322436165124331134152433243124151311412432133431
112335424234263431245116414213344131341333162423321633333413342336114142
111642425126132634352436112542163641263425151663421631563334414224353624
336342344151244211425642554163512441353634121631113224243124121623425126
435116423536341216313413431316312451241311164234125416324216264142113536
243216363334331141423424312415131641422441243213343134132432163333344142
343156263436112332553611231516316316324131341311411334123415335532353634
414236113341421334324251264151244211425635363412163155231126112351242624
331631621263422335424211262611266142343543414234163216414234425126341533
112634331626344234365516442436325524122436222413241563424135361113161531
```

```
241355324142111324425641423424323441425623114255416351434124321334313413
413536341216311132242335424151244211633534413116153324161311223355326131
163216334234322611511641421316333334143413344135362463422463425126413414
311141244216415623354251244211425641312442335525421626414212162316152433
341434353634354341261123354233242642343316124315164242512633341234315652
243341421343334322331146163311231142554163514323331126341312162335363412
16313413425126
```

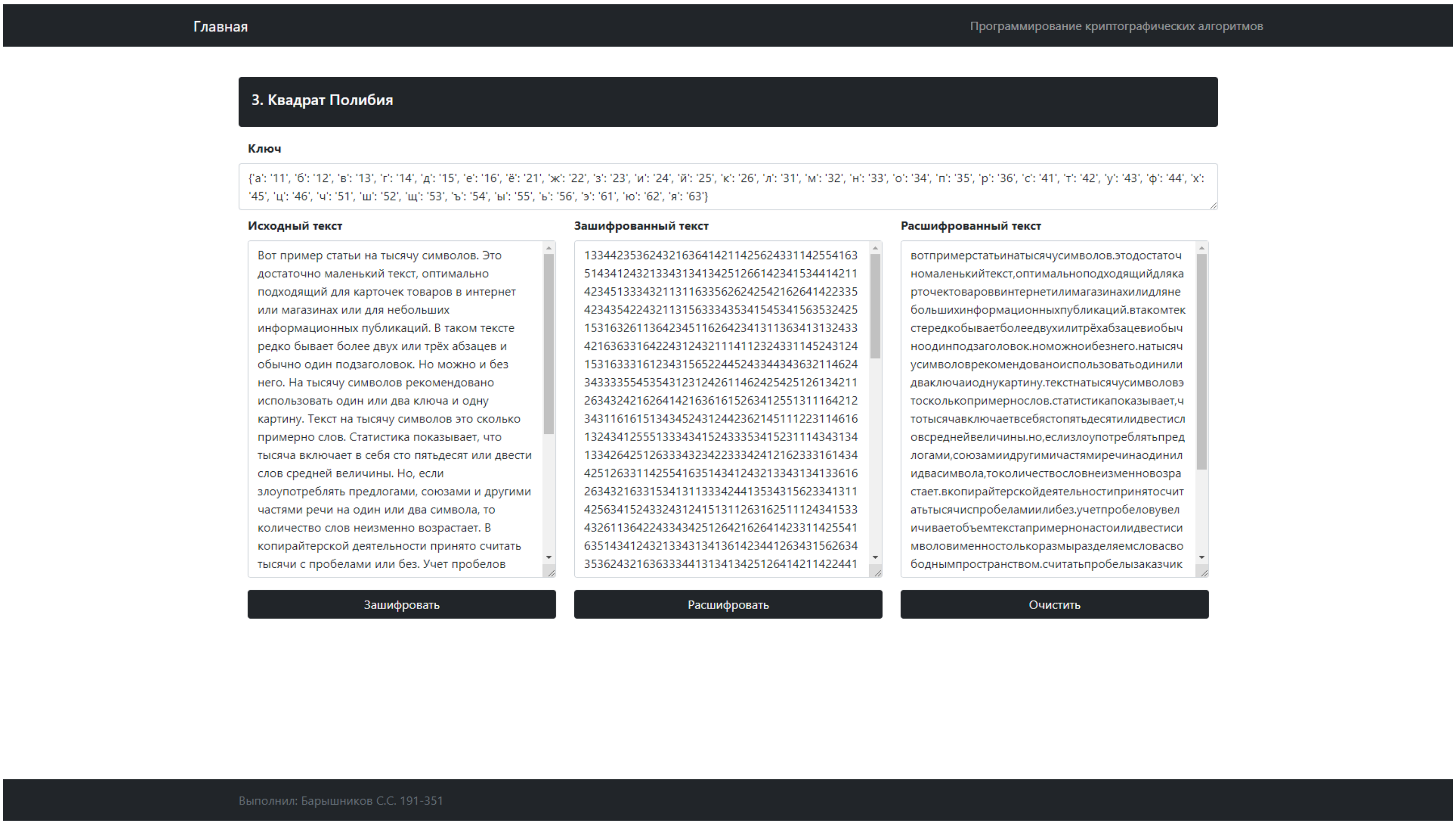
Расшифрованный текст:

```
вотпримерстатьинатысячусимволов.этодостаточномаленькийтекст,оптимальнопо
дходящийдлякарточектовароввинтернетилимагазинахилидлянебольшихинформацио
нныхпубликаций.втакомтекстередкобываетболеедвухилитрёхабзацевиобычноодинп
одзаголовок.номожноибезнего.натысячусимволоврекомендованоиспользоватьоди
нилидваключаиоднукартину.текстнатысячусимволовэтосколькопримернослов.ста
тистикапоказывает,чтотысячавключаетвсебястопятьдесятилидвестисловсредней
величины.но,еслизлоупотреблятьпредлогами,союзамиидругимичастямиречинаоди
нилидвасимвола,токоличествословнеизменновозрастает.вкопирайтерскойдеятел
ьностипринятосчитатьтысячиспробеламиилибез.учетпробеловувеличиваетобъемт
екстапримернонастоилидвестисимволовименностолькоразмыразделяемсловасвобо
днымпространством.считатьпробелызаказчикинелюбят,таккакэтопустоеместо.од
наконекоторыефирмыибиржевидятсправедливымставитьстоимостьзатысячусимволо
вспробелами,считаяпоследниеважнымэлементомкачественноговосприятия.соглас
итесь,читатьслитныйтекстбезединогопропуска,никтонебудет.нобольшинствунуж
наценазатысячузнаковбезпробелов.
```

Карточка:



Интерфейс:


Главная
Программирование криптографических алгоритмов
3. Квадрат Полибия
Ключ
{'а': '11', 'б': '12', 'в': '13', 'г': '14', 'д': '15', 'е': '16', 'ё': '21', 'ж': '22', 'з': '23', 'и': '24', 'й': '25', 'к': '26', 'л': '31', 'м': '32', 'н': '33', 'о': '34', 'п': '35', 'р': '36', 'с': '41', 'т': '42', 'у': '43', 'ф': '44', 'х': '45', 'ц': '46', 'ч': '51', 'ш': '52', 'щ': '53', 'ъ': '54', 'ы': '55', 'ь': '56', 'э': '61', 'ю': '62', 'я': '63'}
Исходный текст
Вот пример статьи на тысячу символов. Это достаточно маленький текст, оптимально подходящий для карточек товаров в интернет или магазинах или для небольших информационных публикаций. В таком тексте редко бывает более двух или трёх абзацев и обычно один подзаголовок. Но можно и без него. На тысячу символов рекомендовано использовать один или два ключа и одну картину. Текст на тысячу символов это сколько примерно слов. Статистика показывает, что тысяча включает в себя сто пятьдесят или двести слов средней величины. Но, если злоупотреблять предлогами, союзами и другими частями речи на один или два символа, то количество слов неизменно возрастает. В копирайтерской деятельности принято считать тысячи с пробелами или без. Учет пробелов
Зашифрованный текст
1334423536243216364142114256243311425541635143412432133431341342512661423415344142114234513334321131163356262425421626414223354234354224321131563334353415453415635324251531632611364234511626423413113634131324334216363316422431243211141123243311452431241531633316123431565224452433443436321146243433335545354312312426114624254251261342112634324216264142163616152634125513111642123431161615134345243124423621451112231146161324341255513334341524333534152311143431341334264251263334323422333424121623331614344251263311425541635143412432133431341336162634321633153413113334244135343156233413114256341524332431241513112631625111243415334326113642243343425126421626414233114255416351434124321334313413614234412634315626343536243216363334413134134251264142114224 41
Расшифрованный текст
вотпримерстатьинатысячусимволов.этодостаточномаленькийтекст,оптимальноподходящийдлякарточектовароввинтернетилимагазинахилидлянебольшихинформационныхпубликаций.втакомтекстередкобываетболеедвухилитрёхабзацевиобычноодинподзаголовок.номожноибезнего.натысячусимволоврекомендованоиспользоватьодинилидваключаиоднукартину.текстнатысячусимволовэтосколькопримернослов.статистикапоказывает,чтотысячавключаетвсебястопятьдесятилидвестисловсреднейвеличины.но,еслизлоупотреблятьпредлогами,союзамиидругимичастямиречинаодинилидвасимвола,токоличествословнеизменновозрастает.вкопирайтерскойдеятельностипринятосчитатьтысячиспробеламиилибез.учетпробеловувеличиваетобъемтекстапримернонастоилидвестисимволовименностолькоразмыразделяемсловасвободнымпространством.считатьпробелызаказчик
Зашифровать
Расшифровать
Очистить
Выполнил: Барышников С.С. 191-351

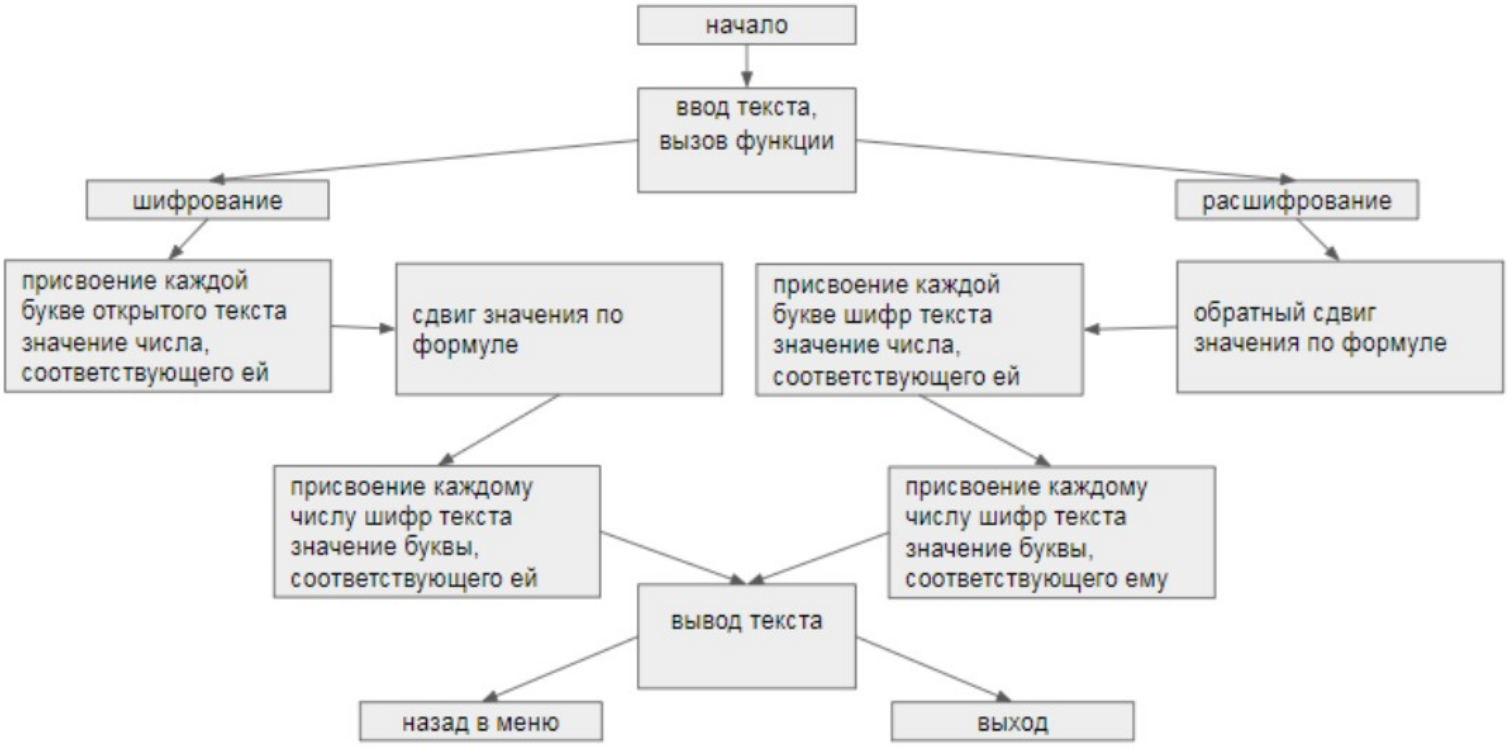
Блок В: ШИФРЫ МНОГОЗНАЧНОЙ ЗАМЕНЫ

4. Шифр Тритемия

Шифр Тритемия предполагал использование алфавитной таблицы. Он использовал эту таблицу для многоалфавитного зашифрования самым простым из возможных способов: первая буква текста шифруется первым алфавитом, вторая буква — вторым и т. д. В этой таблице не было отдельного алфавита открытого текста, для этой цели служил алфавит первой строки. Таким образом, открытый текст, начинающийся со слов HUNC CAVETO VIRUM ..., приобретал вид HXPF GFBMCZ FUEIB ... .

Преимущество этого метода шифрования по сравнению с методом Альберти состоит в том, что с каждой буквой задействуется новый алфавит. Альберти менял алфавиты лишь после трех или четырех слов. Поэтому его шифртекст состоял из отрезков, каждый из которых обладал закономерностями открытого текста, которые помогали вскрыть криптограмму. Побуквенное зашифрование не дает такого преимущества. Шифр Тритемия является также первым нетривиальным примером периодического шифра. Так называется многоалфавитный шифр, правило зашифрования которого состоит в использовании периодически повторяющейся последовательности простых замен.

Блок-схема:



Код программы:

```
# импорт компонентов, необходимых для работы программы
from base import alphabet, input_for_cipher_short,
input_for_cipher_long, output_from_decrypted

# функция расшифрования
def trithemius_decode(input):
```

```
    decode: str = ""
    k = 0
    for position, symbol in enumerate(input):
        index = (alphabet.find(symbol) + k) % len(alphabet)
        decode += alphabet[index]
        k -= 1
    return decode

# функция шифрования
def trithemius_encode(input):
    encode = ""
    k = 0
    for position, symbol in enumerate(input):
        index = (alphabet.find(symbol) + k) % len(alphabet)
        encode += alphabet[index]
        k += 1
    return encode

#вывод результатов работы программы
print(f'''
Шифр Тритемия:
КОРОТКИЙ ТЕКСТ:
Зашифрованный текст:
{trithemius_encode(input_for_cipher_short())}

Расшифрованный текст:
{output_from_decrypted(trithemius_decode(trithemius_encode(
    input_for_cipher_short())))}

ДЛИННЫЙ ТЕКСТ:
Зашифрованный текст:
{trithemius_encode(input_for_cipher_long())}

Расшифрованный текст:
{output_from_decrypted(trithemius_decode(trithemius_encode(
    input_for_cipher_long())))}
''')
```

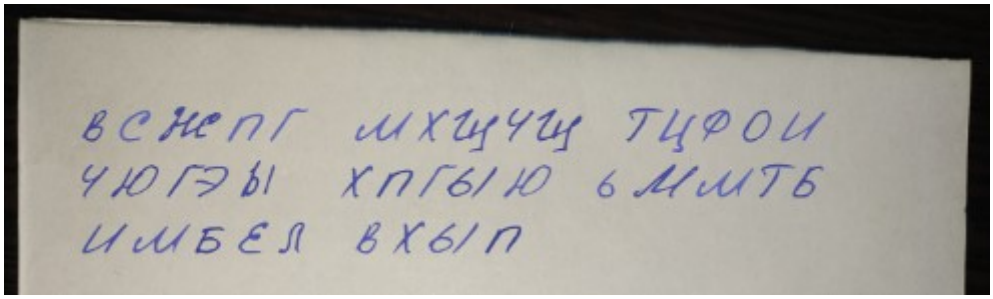
Тестирование:

```
/bin/python3
/root/mospolytech-education-crypt-dev-2021-1/lab02_4_trithemius.py

Шифр Тритемия:
КОРОТКИЙ ТЕКСТ:
Зашифрованный текст:
всжпгмхщчщтцфоичюгэыхпгыюьммтбимбелвхып

Расшифрованный текст:
время,приливыиотливынеждутчеловека.

ДЛИННЫЙ ТЕКСТ:
Зашифрованный текст:
```

**впфтфнтлшъькюицьпгмдтлизаеыижкярцкюфсзучщзышвщыъоыхяоюэяиьгкмглпмотйогпбухччнаърмзшъхютяхжйжряёолаярпдемтшлшцоэфшцыпусъьвхладвюжыкгаомюымъофьчъчлгцээюмзгзцвагщрдёпхйвувнтсшлтъьпьссщюсмфущзчёдюяяюрузлйфуйъёзпиапнхьпкзябвшюджжэвыялйнвпмхыпухфчршъхоучюцхвжмбешлхмыфсриндспузчмушчръсэсрябьёеегфбиэьпъндйплпнйизухигмцэуюеэуезяеллсёовиртовхяцеюыътчныщэсндбеядвугзйлейгпнпуотжешъиьуэяцщпааэуьршчэлвкофрнтьувыезсужбкряпафсрдёгехйфэямыпжкиедзхчошучльъссьфъучяэмяеулсёйлёотёуоммсхышшэьсоъогнвдщщвёыщжддмррояйгрнокшьмушхеобгряьеънаёщияекжкиедгхнтицфйтыяаэъяэыкщжищжкёйюкгззнжрсузхпщйьюмтбвзьфюгеязшгмамоинйежвцйсгхчыэфьючэпдбелюичкхмцньхзсртсейсжфстцфнтцвзьёкщзёзжъяжкясбемкъмжёлъчкерщауъвдтгеюгьижииьорнимкжёйчщрътчныщэсщючбвдзйжецёкнюжмьтгликтжнцьчыпобтаувшгсдзйимонюмсудсрчэсщатэляйюаятчяпэцвшбсджёлдлягвцмхщъифхлчкбюаедёгъгьйлинузмгнбмссйрхъчьёрбцяьлъуьеьящбэщннмоявёёомжбущыймфяйавяъявёзтяшлхмыфсриндспузшчщшлршнышшбеелюиччиимзввмиёдгктбуефочръаэъысйшёзшьёжиюкямрпсквцнещяуыщошнашгцдпъеиоблъншойзтоэмцйршъйотррцьюуавдгейигщвкокжйтппзешйлъыбхщыоэымйыбёзшкёмбиьугмаивхяцеюырчкцыппфшбгвхъвёълсё

Расшифрованный текст:
вотпримерстатьинатысячусимволов.этодостаточномаленькийтекст,оптимальнопо
дходящийдлякарточектовароввинтернетилимагазинахилидлянебольшихинформацио
нныхпубликаций.втакомтекстередкобываетболеедвухилитрёхабзацевиобычноодинп
одзаголовок.номожноибезнего.натысячусимволоврекомендованоиспользоватьоди
нилидваключаиоднукартину.текстнатысячусимволовэтосколькопримернословяста
тистикапоказывает,чтотысячавключаетвсебястопятьдесятилидвестисловсредней
величины.но,еслизлоупотреблятьпредлогами,союзамиидругимичастямиречинаоди
нилидвасимвола,токоличествословнеизменновозрастает.вкопирайтерскойдеятел
ьностипринятосчитатьтысячиспробеламиилибез.учетпробеловувеличиваетобъемт
екстапримернонастоилидвестисимволовименностолькоразмыразделяемсловасвобо
днымпространством.считатьпробелызаказчикинелюбят,таккакэтопустоеместо.од
наконекоторыефирмыибиржividятсправедливымставитьстоимостьзатысячусимволо
вспробелами,считаяпоследниеважнымэлементомкачественноговосприятия.соглас
итесь,читатьслитныйтекстбезединогопропуска,никтонебудет.нобольшинствунуж
наценазатысячузнаковбезпробелов.

Карточка:



Интерфейс:

4. Шифр Тритемия

Исходный текст

Вот пример статьи на тысячу символов. Это достаточно маленький текст, оптимально подходящий для карточек товаров в интернет или магазинах или для небольших информационных публикаций. В таком тексте редко бывает более двух или трёх абзацев и обычно один подзаголовок. Но можно и без него. На тысячу символов рекомендовано использовать один или два ключа и одну картину. Текст на тысячу символов это сколько примерно слов. Статистика показывает, что тысяча включает в себя сто пятьдесят или двести слов средней величины. Но, если злоупотреблять предлогами, союзами и другими частями речи на один или два символа, то количество слов неизменно возрастает. В копирайтерской деятельности принято считать тысячи с пробелами или без. Учет пробелов увеличивает объем текста примерно на сто или двести символов именно столько раз мы

Зашифрованный текст

внрммджюииижпыярвйялгюысфифртефшкьрлайллшйемввясьфыйчффьнтшшмухрртэкэзчзззёъкгчскщщтщлцкъыцюкпцсдбрнаяеимюиеыззяяыюртоххщцкясупйргргжвокъхергжмёжвфйыаюэйгьялхстжьннхщлвсюзкзмюгиияъешцьярйпмрэкчулкиёцчйлзрнвръъачмъцьбтлжьыыпучшцкнежснпвнипуёззеёзггьуцшзурыювфцзщбцгъхтилалзйьйэвевщбцатпьыхэъшфдофждхюпдзлёздюышввулуыафэвшмьэтцыщзпцзмттмюпчмщрлиявцвюатмбьюцщхетххxмпжснннпикэмргиижьддщъоэыцксдйжкчлтфштнршнъсщъввулучдтбуолээшшжщвиифбуикжбюаллбивдцегцфэучорцтарфбчыннппзорвнжгбдёйегёгцсынаиыыомухиерлйртрмыдыжвбыжичьяъзпаалшуъняпуетжкоикжбюылвещебфывдгющьшфвоъъифцпсдоезёйбиззъёюёфеесхвбечнхшшрчёоцзттknзббъмюдуддёёыббшьмяъьбсъжшбцюуачзпммйыюдчгюзяыпхчбечявоыччфёиоргубгиетгышьидхнчюгфщяялъъсфлцттрвттнжиеяьюййяжэафаьюрцщршччщщфранрсажкшмывээгхъбеюатпапымщнцгуххтффсамппюйжлпвзмэётдмяяьнрцерзс

Расшифрованный текст

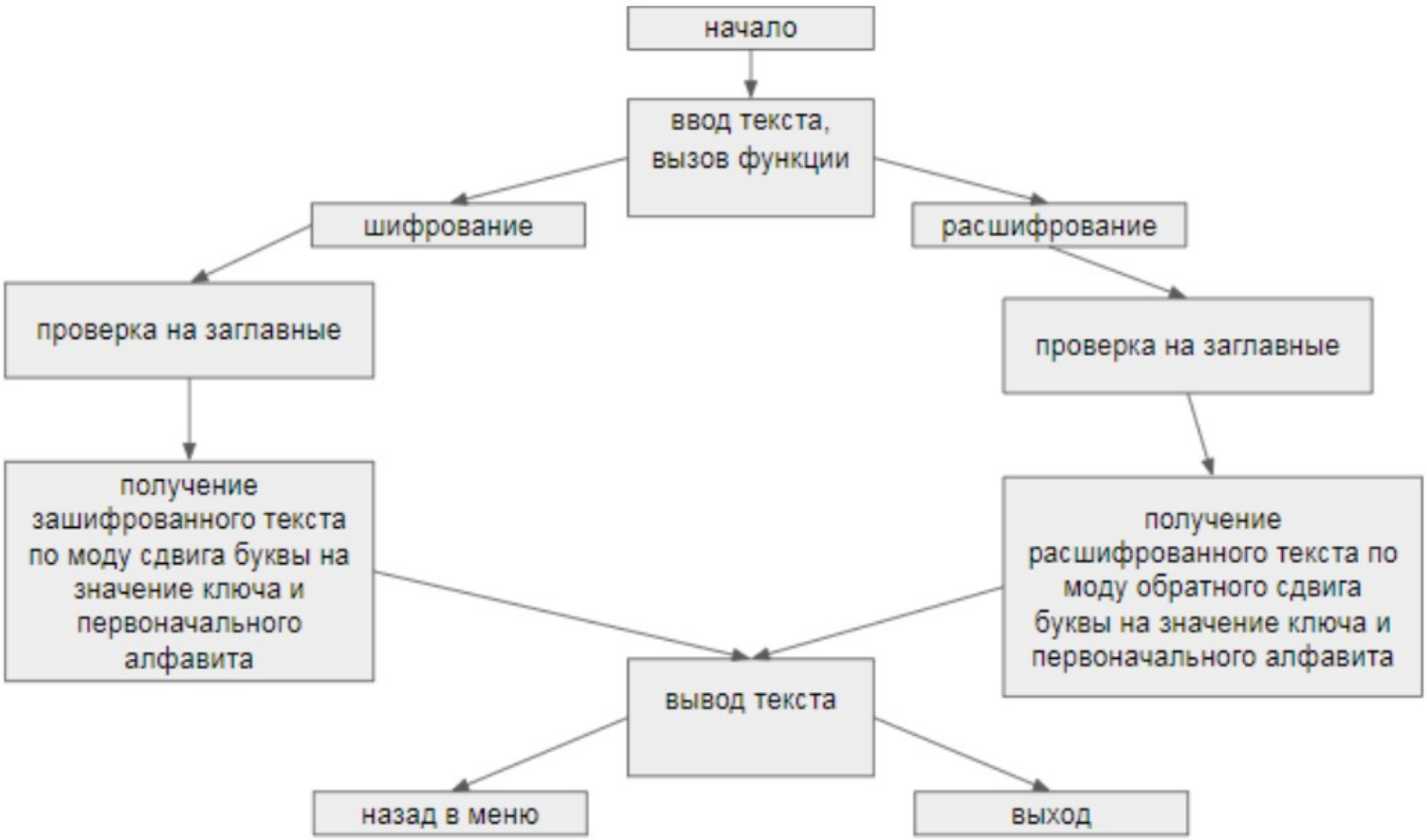
вотпримерстатьинатысячусимволов.этодостаточномаленькийтекст,оптимальноподходящийдлякарточектовароввинтернетилимагазинахилидляне большихинформационныхпубликаций.втакомтекстередкобываетболеедвухилитрёхабзацевиобычноодинподзаголовок.номожноибезнего.натысячусимволоврекомендованоиспользоватьодинилидваключаионуюкартину.текстнатысячусимволовэтосколькопримерно слов.статистикапоказывает,чтотысячавключаетвсебястопятьдесятилидвестисловсреднейвеличины.но,еслизлоупотреблятьпредлогами,союзамиидругимичастямиречинаодинилидвасимвола,токолическтвословнеизменновозрастает.вкопирайтерскойдеятельностипринятосчитатьтысячиспробеламиилибез.учетпробеловувеличиваетобъемтекстапримернонастоилидвестисимволовименностолькоразмыразделяемсловасвободнымпространством.считатьпробелызаказчикинелюбят,таккакэтопустоеместо.однаконекоторыефирмыибиржividятсправедливымставитьсто

Зашифровать | Расшифровать | Очистить

5. Шифр Белазо

В 1553 Джованни Баттиста Белазо предложил использовать для многоалфавитного шифра буквенный, легко запоминаемый ключ, который он назвал паролем. Паролем могло служить слово или фраза. Пароль периодически записывался над открытым текстом. Буква пароля, расположенная над буквой текста, указывала на алфавит таблицы, который использовался для зашифрования этой буквы. Например, это мог быть алфавит из таблицы Тритемия, первой буквой которого являлась буква пароля. Однако Белазо, как и Тритемий, использовал в качестве алфавитов шифра обычные алфавиты.

Блок-схема:



Код программы:

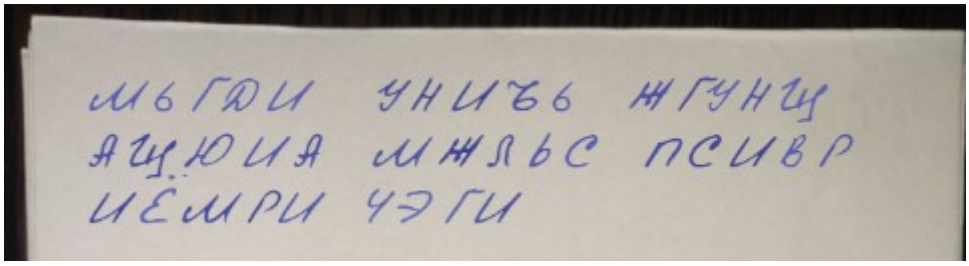
```
# импорт компонентов, необходимых для работы программы
from base import alphabet, input_for_cipher_short,
input_for_cipher_long, output_from_decrypted

# установка ключа
key = 'ключ'

# функция расшифрования
def bellaso_decode(input, key):
    decrypted = ''
    offset = 0
    for ix in range(len(input)):
        if input[ix] not in alphabet:
            output = input[ix]
```

```
            offset += -1
        elif (alphabet.find(input[ix])) > (len(alphabet) -
(alphabet.find(key[((ix + offset) % len(key))]))) - 1):
            output = alphabet[(alphabet.find(
                input[ix]) - (alphabet.find(key[((ix + offset) %
len(key))])))) % 33]
        else:
            output = alphabet[alphabet.find(
                input[ix]) - (alphabet.find(key[((ix + offset) %
len(key))]))))]
        decrypted += output
    return decrypted

# функция шифрования
def bellaso_encode(input, key):
    encoded = ''
    offset = 0
    for ix in range(len(input)):
        if input[ix] not in alphabet:
            output = input[ix]
            offset += -1
        elif (alphabet.find(input[ix])) > (len(alphabet) -
(alphabet.find(key[((ix + offset) % len(key))]))) - 1):
            output = alphabet[(alphabet.find(
                input[ix]) + (alphabet.find(key[((ix + offset) %
len(key))])))) % 33]
        else:
            output = alphabet[alphabet.find(
                input[ix]) + (alphabet.find(key[((ix + offset) %
len(key))]))))]
        encoded += output
    return encoded

#вывод результатов работы программы
print(f'''
Шифр Белазо:
Ключ: {key}
КОРОТКИЙ ТЕКСТ:
Зашифрованный текст:
{bellaso_encode(input_for_cipher_short(), key)}

Расшифрованный текст:
{output_from_decrypted(bellaso_decode(bellaso_encode(
    input_for_cipher_short(), key), key))}

ДЛИННЫЙ ТЕКСТ:
Зашифрованный текст:
{bellaso_encode(input_for_cipher_long(), key)}

Расшифрованный текст:
{output_from_decrypted(bellaso_decode(bellaso_encode(
    input_for_cipher_long(), key), key))}
''')
```

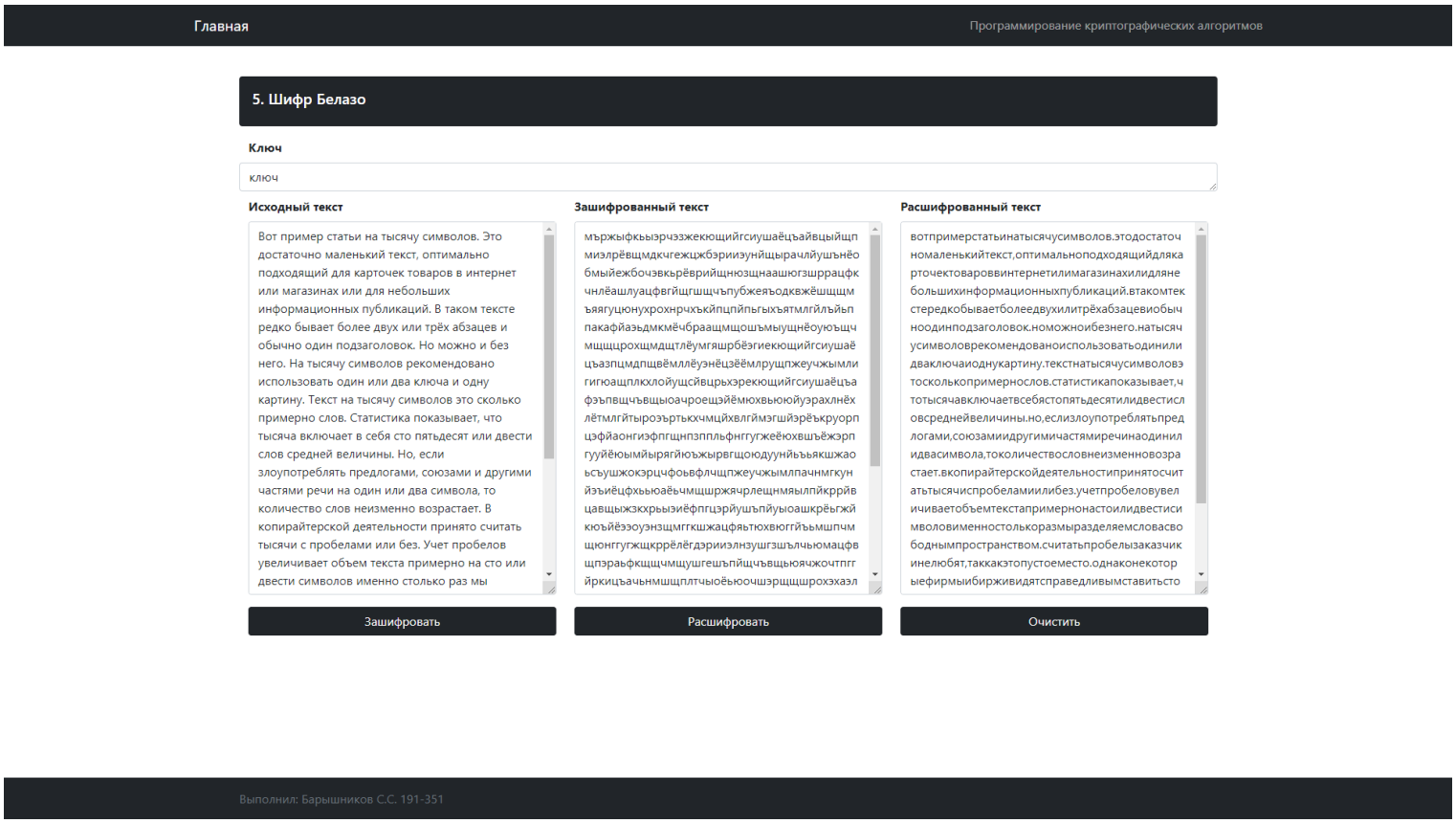
Тестирование:

```
/bin/python3
/root/mospolytech-education-crypt-dev-2021-1/lab02_5_bellaso.py

Шифр Белазо:
Ключ: ключ
КОРОТКИЙ ТЕКСТ:
Зашифрованный текст:
мьгдйунйъьжгунщащюйамжльспсйврйёмричэги

Расшифрованный текст:
время,приливыиотливынеждутчеловека.

ДЛИННЫЙ ТЕКСТ:
Зашифрованный текст:
мържыфкьыэрчэзжекющийгсиушаёцъайвцыйщпмиэлрёвщмдкчгежцжбэриизунйщырачлйуш
ънёобмыйежбочэвкьрёврийщнюзщнаашюгзшррацфкчнлёашлуацфвгйщгшщчъпубжеяъодкв
жёшщшмъяягуцюнухрохнрчхъкйпцпйпьгыхъятмлгйлъйьппакафйаэьдмкмёчбраащмщошъм
ыущнёоуюъщчмщщцрохщмдштлёумгяшрбёэгиекющийгсиушаёцъазпцмдпщвёмллёуэнёцзёё
млрущпжеучжымлигигюащплкхлойущсйвцрьхэрекющийгсиушаёцъафэъпвщчъвщыоачроещ
эйёмюхвьююйуэрахлнёхлётмлгйтыроэъртъкхчмцйхвлгймэгшйэрёъкруорпцэфйаонгиэф
пгщнпзппльфнггугжеёюхвшъёжэрпгууйёюымйырягйюъжырвгщоюдуунйьъьякшжаоьсъушж
окэрцчфоьвфлчщпжеучжымлпачнмгкунйэъиёцфхььюаёьчмщшржячрлещнмяылпйкррйвцав
щыжзкхрыэиёфпгцэрйушъпйуыоашкрёъгжйкюъйёээоуэнзщмггкшжацфяьтюхвюггйъъмшп
чмщюнггугжщкррёлёгдэрииэлнзушгзшълчьюмацфвщпэраьфкщщчмщушгешъпйщчъвщьюячж
очтпггйркицъачьнмшщплтчыоёьюочшэрщщшрохэхаэлруъьмшпчщякцюявфиашрйхлкряъюр
чхцювзюмжюэрёпшгиэърохъвекцмепцмйщьщьяфодёфяаытжщупэйъыочмрвгунщдьююющуюъи
эъждщэрутлртъкххкьфкщщчмщъыоёлрйччфёжээхаэлэжщэйъощжьмлееёшыгпшгеэъквкггиэ
нгешъбёмъпжыфэйукрохэмъцлпаэрпутыроуююйжэйаэщщбэриизмгяппжещомжыънкьцюятю
лахюмепмсыпюрохщмшщчъпущпймялксщюнпщюякющийгсяшлиёммгяъьмшпчмщэги

Расшифрованный текст:
вотпримерстатьинатысячусимволов.этодостаточномаленькийтекст,оптимальнопо
дходящийдлякарточектовароввинтернетилимагазинахилидлянебольшихинформацио
нныхпубликаций.втакомтекстередкобываетболеедвухилитрёхабзацевиобычноодинп
одзаголовок.номожноибезнего.натысячусимволоврекомендованоиспользоватьоди
нилидваключаиоднукартину.текстнатысячусимволовэтосколькопримернослов.ста
тистикапоказывает,чтотысячавключаетвсебястопятьдесятилидвестисловсредней
величины.но,еслизлоупотреблятьпредлогами,союзамиидругимичастямиречинаоди
нилидвасимвола,токоличествословнеизменновозрастает.вкопирайтерскойдеятел
ьностипринятосчитатьтысячиспробеламиилибез.учетпробеловувеличиваетобъемт
екстапримернонастоилидвестисимволовименностолькоразмыразделяемсловасвобо
днымпространством.считатьпробелызаказчикинелюбят,таккакэтопустоеместо.од
наконекоторыефирмыибиржividятсправедливымставитьстоимостьзатысячусимволо
вспробелами,считаяпоследниеважнымэлементомкачественноговосприятия.соглас
итесь,читатьслитныйтекстбезединогопропуска,никтонебудет.нобольшинствунуж
наценазатысячузнаковбезпробелов.
```

Карточка:



Интерфейс:

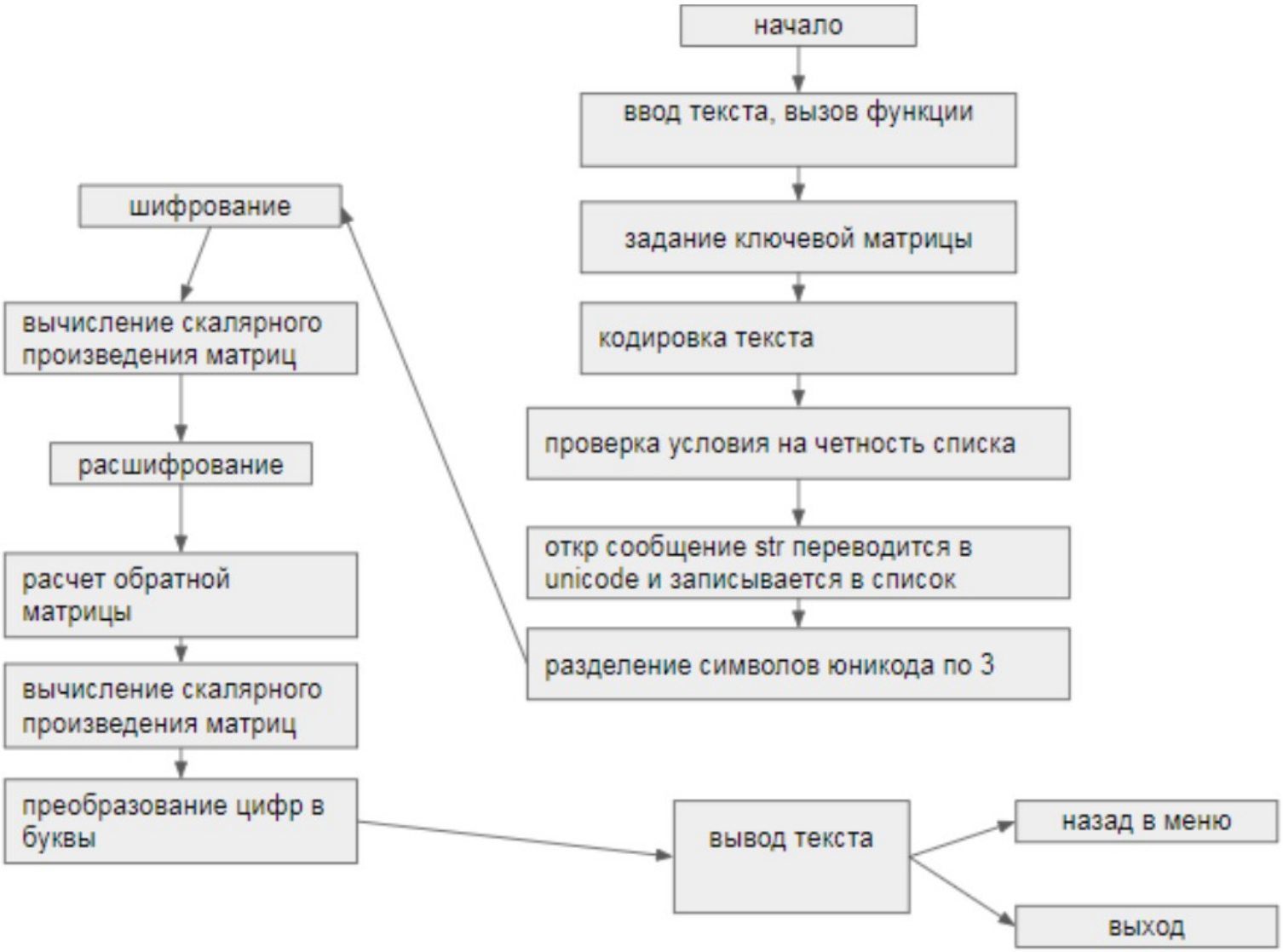

Главная
Программирование криптографических алгоритмов
5. Шифр Белазо
Ключ
ключ
Исходный текст
Вот пример статьи на тысячу символов. Это достаточно маленький текст, оптимально подходящий для карточек товаров в интернет или магазинах или для небольших информационных публикаций. В таком тексте редко бывает более двух или трёх абзацев и обычно один подзаголовок. Но можно и без него. На тысячу символов рекомендовано использовать один или два ключа и одну картину. Текст на тысячу символов это сколько примерно слов. Статистика показывает, что тысяча включает в себя сто пятьдесят или двести слов средней величины. Но, если злоупотреблять предлогами, союзами и другими частями речи на один или два символа, то количество слов неизменно возрастает. В копирайтерской деятельности принято считать тысячи с пробелами или без. Учет пробелов увеличивает объем текста примерно на сто или двести символов именно столько раз мы
Зашифрованный текст
мържыфкыэрчззжекющийгсиушаёцъайвцыйщпмиэлрёвщмдкчгежцжбэриизунйщырачлйушънёобмыйёжбочзвкьрёврийщнюзщнаашюгзшррацфкчнлёашлуацфвгйщгшщчъпубжеяъодквжёшщщмъяягуцюнухрохнрчхъкйпцпйпыгыхъятмлгйлъйьппакафйаэьдмкмёчбраащмщошъмыущнёоуюъщчмщщцрохщмдщтлёумгяшрбёзгиекющийгсиушаёцъазпцмдпщвёмллёуэнёцзёёмлрущпжеучжымлигигюащплкхлойущсйвцрьхэрекющийгсиушаёцъафэъпвщчъвщыоачроещэйёмюхвьююйуэрахлнёхлётмлгйтыроэъртькхчмцйхвлгймэгшйэрёъкруорпцэфйаонгизфпгщнпзппльфнггугжеёюхвшъёжэрпгууйёюымйырягйюъжырвгщоюдуунйьъякшжаоьсъушжокэрцчфоьвфлчщпжеучжымлпачнмгкунйэъиёцфхьыюаёьчмщшржячрлещнмяылпйкррйвцавщыжэкхрьыэиёфпгцэрйушъпйуыоашкрёьгжйкюъйёээоуэнзщмггкшжацфяьтюхвюггйъьмшпчмщюнггугжщкррёлёгдэриизлнзушгзшъльчьюмацфвщпэраьфкщщчмщушгешъпйщчъвщыюячжочтпггйркицъачьнмшщплтчыоёьюочшэрщщшрохэхаэл
Расшифрованный текст
вотпримерстатьинатысячусимволов.этодостаточномаленькийтекст,оптимальноподходящийдлякарточектовароввинтернетилимагазинахилидляне большихинформационныхпубликаций.втакомтекстередкобываетболеедвухилитрёхабзацевиобычноодинподзаголовок.номожноибезнего.натысячусимволоврекомендованоиспользоватьодинилидваключаиоднукартину.текстнатысячусимволовэтосколькопримернослов.статистикапоказывает,чтотысячавключаетвсебястопятьдесятилидвестисловсреднейвеличины.но,еслизлоупотреблятьпредлогами,союзамиидругимичастямиречинаодинилидвасимвола,токоличествословнеизменновозрастает.вкопирайтерскойдеятельностипринятосчитатьтысячиспробеламиилибез.учетпробеловувеличиваетобъемтекстапримернонастоилидвестисимволовименностолькоразмыразделяемсловасвободнымпространством.считатьпробелызаказчикинелюбят,таккакэтопустоеместо.однаконекоторыефирмыибирживидятсправедливымставитьсто
Зашифровать
Расшифровать
Очистить
Выполнил: Барышников С.С. 191-351

Блок С: ШИФРЫ БЛОЧНОЙ ЗАМЕНЫ

8. Матричный шифр

Шифр Хилла — полиграммный шифр подстановки, основанный на линейной алгебре и модульной арифметике. Изобретён американским математиком Лестером Хиллом в 1929 году. Это был первый шифр, который позволил на практике (хотя и с трудом) одновременно оперировать более чем с тремя символами. Шифр Хилла не нашёл практического применения в криптографии из-за слабой устойчивости ко взлому и отсутствия описания алгоритмов генерации прямых и обратных матриц большого размера.

Блок-схема:



Код программы:

```
# импорт компонентов, необходимых для работы программы
from base import alphabet, input_for_cipher_short,
input_for_cipher_long, output_from_decrypted
import numpy as np
from egcd import egcd
```

```
# установка ключа
key = '3 10 20 20 19 17 23 78 17'
inp = key.split(' ')

key = np.matrix([[int(inp[0]), int(inp[1]), int(inp[2])], [int(inp[3]),
int(
    inp[4]), int(inp[5])], [int(inp[6]), int(inp[7]), int(inp[8])]])

letter_to_index = dict(zip(alphabet, range(len(alphabet))))
index_to_letter = dict(zip(range(len(alphabet)), alphabet))

# функция вычисления обратной матрицы
def matrix_mod_inv(matrix, modulus):
    det = int(np.round(np.linalg.det(matrix)))
    det_inv = egcd(det, modulus)[1] % modulus
    matrix_modulus_inv = (
        det_inv * np.round(det * np.linalg.inv(matrix)).astype(int) %
modulus
    )

    return matrix_modulus_inv

# функция шифрования
def matrix_encode(message, K):
     # проверка на определитель равный 0
     if np.linalg.det(K) == 0:
          raise ValueError('Определитель матрицы равен 0! Дальнейшая
работа программы невозможна!')

    encrypted = ""
    message_in_numbers = []
    for letter in message:
        message_in_numbers.append(letter_to_index[letter])

    split_P = [
        message_in_numbers[i: i + int(K.shape[0])]
        for i in range(0, len(message_in_numbers), int(K.shape[0]))
    ]

    for P in split_P:
        P = np.transpose(np.asarray(P))[:, np.newaxis]

        while P.shape[0] != K.shape[0]:
            P = np.append(P, letter_to_index[" "])[:, np.newaxis]

        numbers = np.dot(K, P) % len(alphabet)
        n = numbers.shape[0]

        for idx in range(n):
            number = int(numbers[idx, 0])
            encrypted += index_to_letter[number]
    return encrypted
```

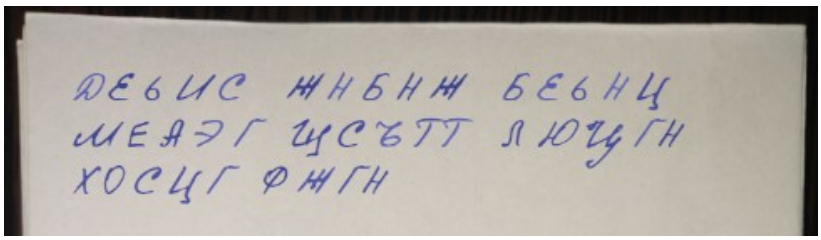
```
# функция расшифрования
def matrix_decode(cipher, Kinv):
    decrypted = ""
    cipher_in_numbers = []
    for letter in cipher:
        cipher_in_numbers.append(letter_to_index[letter])

    split_C = [
        cipher_in_numbers[i: i + int(Kinv.shape[0])]
        for i in range(0, len(cipher_in_numbers), int(Kinv.shape[0]))
    ]

    for C in split_C:
        C = np.transpose(np.asarray(C))[:, np.newaxis]
        numbers = np.dot(Kinv, C) % len(alphabet)
        n = numbers.shape[0]

        for idx in range(n):
            number = int(numbers[idx, 0])
            decrypted += index_to_letter[number]
    return decrypted

#вывод результатов работы программы
print(f'''
Матричный шифр:
Ключ: {key}
КОРОТКИЙ ТЕКСТ:
Зашифрованный текст:
{matrix_encode(input_for_cipher_short(), key).replace(' ', '')}

Расшифрованный текст:
{output_from_decrypted(matrix_decode(matrix_encode(
    input_for_cipher_short(), key), matrix_mod_inv(key,
len(alphabet)))).replace(' ', '')}

ДЛИННЫЙ ТЕКСТ:
Зашифрованный текст:
{matrix_encode(input_for_cipher_long(), key).replace(' ', '')}

Расшифрованный текст:
{output_from_decrypted(matrix_decode(matrix_encode(
    input_for_cipher_long(), key), matrix_mod_inv(key,
len(alphabet)))).replace(' ', '')}
''')
```

Тестирование:

```
/bin/python3
/root/mospolytech-education-crypt-dev-2021-1/lab03_8_matrix.py
Матричный шифр:
Ключ: 3 10 20 20 19 17 23 78 17
КОРОТКИЙ ТЕКСТ:
```

```
Зашифрованный текст:
дёьисжнбнжбеьнцмёаэгщсъттлюцгнхосцгфжгн

Расшифрованный текст:
время,приливыиотливынеждутчеловека.

ДЛИННЫЙ ТЕКСТ:
Зашифрованный текст:
щвичнкфящёжщрээншусзуйдътюцбёэъяшщктыёжщжтийдмпжбзярюмсигфохжртичаужбвфэо
вкквкыкзчвзяжтиьудюяшгмювжаепыэсофрдюейхьёзпфрдчвзйсоюбёъштифчыхвлицхилщн
дкыбокттжжгнщъябллфпыдчикъллвусфвпъбёъётстрэуижлиатйчлихчрозюпдмзмлёъзпжю
цбёввижгнйдмфшжтлйбншедщжгнншусзуйдътюцбёдёьбллющрюяшядмдбмйихюяшбээзмлфр
дяпйшцпнфшшвкающмдуйюьсщигфъарыгхзчэдкхтщтяшцтиоььфчлчнкфящзфэхосжгнёжщюж
ръчасожцповусштдбнивщусзйдзшцпвфшьйотрцачиаъшцъйёгфрдэъпчияъжпшеьёттскиен
аеэёгсюъвуефъацжщофабюязърцпяшлнкчтшюистщоъшхгчкгёядьчтйёьчеажщъёяъьявшзм
лфрдяпйэдкхтщщплгйяоььъжхлжщцэоеяшъбьоиыфэцъаптъхйжгнзыгвядгобомиььхеёжкр
мзпчичнкыфмшэомюцмээусзлеэзнкфмедщкфрдимежгнщцгмфвтэяхосймфыгщмнцьвпёдцщэ
обзчожтвнокдызпмвщвгщзыюжрътюцбёпчръщръмяюбёмллаьпяыогнобрчгюяшажиьгдмжзн
кёчичыдгъяьиммсшмюцфммрэнохюёжюпплкяыщбяьъиохжиъожфцшщъщьачижччачижгншвкв
хшёсошязцезрэлцъсшдыоемъсимзецарцётьнцфжуэфгъмммвщебжюммвфудпулкктиэхосйб
мтэячтфтщогземюцлиьвъжщцкбвжяьбнфжччхлтмъбёючжъяётгъеобэобеежржгнщпгхвдкн
ёыъчдбнмюцъммэуилэчфпыщчишюззмлбасзнкэучцпозбнжяымещкщтжвпуипоэяйщерджбйо
зюьщштщрэжюусзчйдчшрозжншрэнъенэъягнх

Расшифрованный текст:
вотпримерстатьинатысячусимволов.этодостаточномаленькийтекст,оптимальнопо
дходящийдлякарточектовароввинтернетилимагазинахилидлянебольшихинформацио
нныхпубликаций.втакомтекстередкобываетболеедвухилитрёхабзацевиобычноодинп
одзаголовок.номожноибезнего.натысячусимволоврекомендованоиспользоватьоди
нилидваключаиоднукартину.текстнатысячусимволовэтосколькопримернослов.ста
тистикапоказывает,чтотысячавключаетвсебястопятьдесятилидвестисловсредней
величины.но,еслизлоупотреблятьпредлогами,союзамиидругимичастямиречинаоди
нилидвасимвола,токоличествословнеизменновозрастает.вкопирайтерскойдеятел
ьностипринятосчитатьтысячиспробеламиилибез.учетпробеловувеличиваетобъемт
екстапримернонастоилидвестисимволовименностолькоразмыразделяемсловасвобо
днымпространством.считатьпробелызаказчикинелюбят,таккакэтопустоеместо.од
наконекоторыефирмыибиржividятсправедливымставитьстоимостьзатысячусимволо
вспробелами,считаяпоследниеважнымэлементомкачественноговосприятия.соглас
итесь,читатьслитныйтекстбезединогопропуска,никтонебудет.нобольшинствунуж
наценазатысячузнаковбезпробелов.
```

Карточка:



Интерфейс:

8. Матричный шифр

Ключ

3 10 20 20 19 17 23 78 17

Исходный текст

Вот пример статьи на тысячу символов. Это достаточно маленький текст, оптимально подходящий для карточек товаров в интернет или магазинах или для небольших информационных публикаций. В таком тексте редко бывает более двух или трёх абзацев и обычно один подзаголовок. Но можно и без него. На тысячу символов рекомендовано использовать один или два ключа и одну картину. Текст на тысячу символов это сколько примерно слов. Статистика показывает, что тысяча включает в себя сто пятьдесят или двести слов средней величины. Но, если злоупотреблять предлогами, союзами и другими частями речи на один или два символа, то количество слов неизменно возрастает. В копирайтерской деятельности принято считать тысячи с пробелами или без. Учет пробелов увеличивает объем текста примерно на сто или двести символов именно столько раз мы

Зашифрованный текст

щвичнкфящёжщрээнш усзуйдътюцбёзъя
шщктыёжщжтийдмпжбзярюмсигфожрти
чаужбвфзовккквкыкзчвз
яжтиьудюяшгмювжаепыэсофрдюейх
ьёзпфрдчвзйсоюбёъштифч
ыхвлицхилщндкыбокттжжгнщъяблпфпыдчикъ
ллвусфвпъбёъётстрэуижлиатйчлихчрозюпдмзмл
ёъзпжюцбёввижгнйдмфшжтлйбншедщжгннш
усзуйдътюцбёдёьбллющрюяшядмдбмйихюяшбэ
ззмлфрдяпйшцпнфшшвкающмдуйюьсщ игфъа
рыгхзчэдкхтщтяшцтиоььфчлчнкфящзфэхосжгнёж
щюжръчасожцповусшт
дбнивщусзйдзшцпвфшьйотрцачиаъшцъйё гфрдэ
ъпчияъжпшеьёттскиенаеэёгсюъвуефъацжщофаб
юязърцпяшлнкчтшюистщоъшхгчкгёядьчтйёьчеа
жщъёяъьявш
змлфрдяпйэдкхтщщплгйяоььъжллжщцэоеяшъ
бьоиыфэцъаптъхй жгнзыгвя дгобомиььхеёж
крмзпчичнкыфмшзомюцмээусзлеэзнкфмедщкфр
димежгнщцгмфвтэяхосймфыгщмнцьвп

Расшифрованный текст

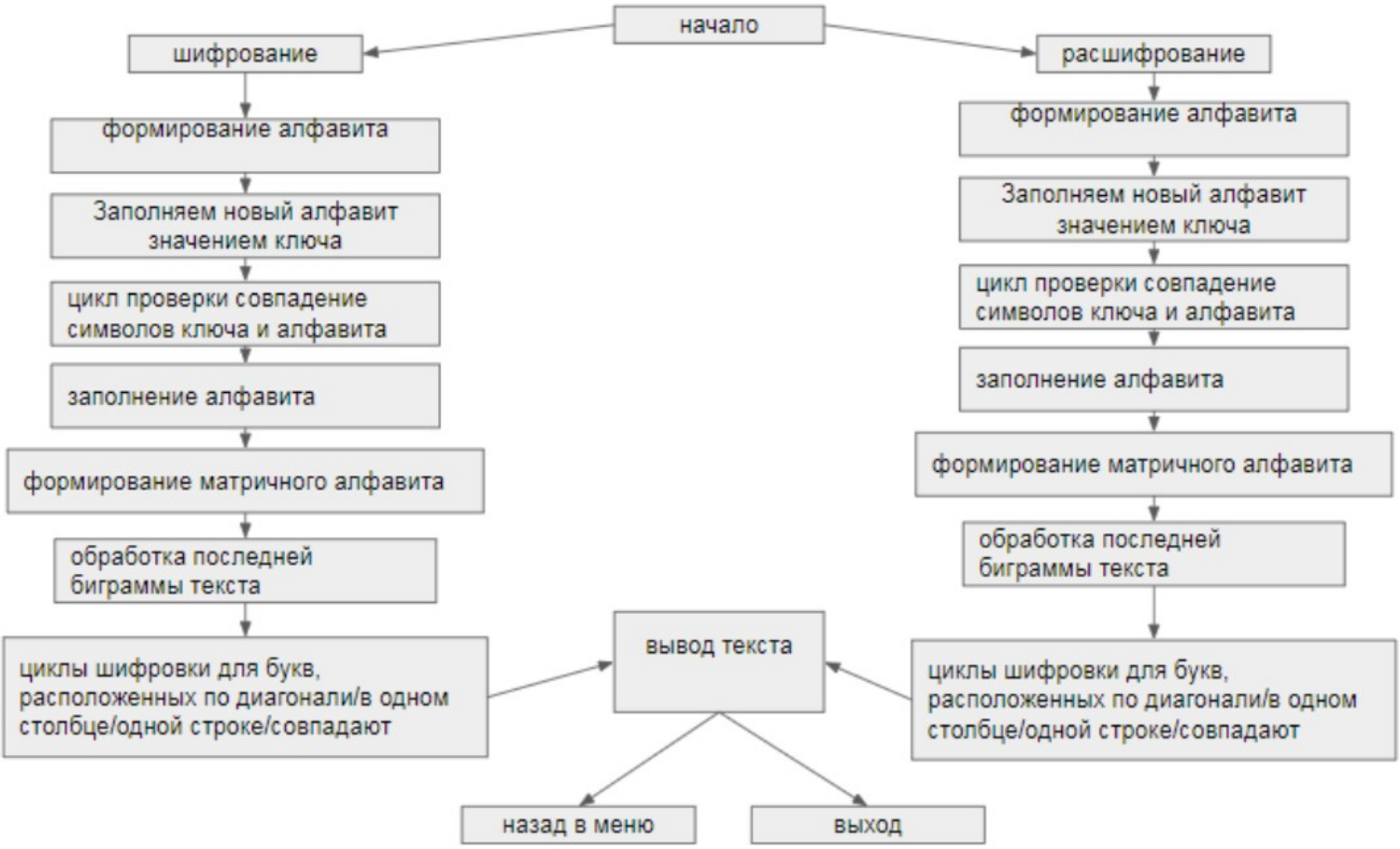
вотпримерстатьинатысячусимволов.этодостаточ
номаленькийтекст,оптимальноподходящийдляка
рточектовароввинтернетилимагазинахилидляне
большихинформационныхпубликаций.втакомтек
стередкобываетболеедвухилитрёхабзацевиобыч
ноодинподзаголовок.номожноибезнего.натысяч
усимволоврекомендованоиспользоватьодинили
двакючаиоднукартину.текстнатысячусимволовэ
тосколькопримернослов.статистикапоказывает,ч
тотысячавключаетвсебястопятьдесятилидвестисл
овсреднейвеличины.но,еслизлоупотреблятьпред
логами,союзамиидругимичастямиречинаодинил
идвасимвола,токолиествословнеизменновозра
стает.вкопирайтерскойдеятельностипринятосчит
атьтысячиспробеламиилибез.учетпробеловувел
ичиваетобъемтекстапримернонастоилидвестиси
мволовименностолькоразмыразделяемсловасво
боднымпространством.считатьпробелызаказчик
инелюбят,таккакэтопустоеместо.однаконекотор
ыефирмыибиржividятсправедливымставитьсто

Зашифровать Расшифровать Очистить

9. Шифр Плейфера

Шифр Плейфера или квадрат Плейфера — ручная симметричная техника шифрования, в которой впервые использована замена биграмм. Изобретена в 1854 году английским физиком Чарльзом Уитстоном, но названа именем лорда Лайона Плейфера, который внёс большой вклад в продвижение использования данной системы шифрования в государственной службе. Шифр предусматривает шифрование пар символов (биграмм) вместо одиночных символов, как в шифре подстановки и в более сложных системах шифрования Виженера. Таким образом, шифр Плейфера более устойчив к взлому по сравнению с шифром простой замены, так как усложняется его частотный анализ. Он может быть проведён, но не для символов, а для биграмм. Так как возможных биграмм больше, чем символов, анализ значительно более трудоёмок и требует большего объёма зашифрованного текста.

Блок-схема:



Код программы:

```
# импорт компонентов, необходимых для работы программы
from base import alphabet, input_for_cipher_short,
input_for_cipher_long, output_from_decrypted
alphabet = alphabet.replace(' ', '') + 'abc'

# установка ключа
key = 'ключ'

# функция шифрования
def playfair_encode(clearText, key):
```

```
    text = clearText
    new_alphabet = []
    for i in range(len(key)):
        new_alphabet.append(key[i])
    for i in range(len(alphabet)):
        bool_buff = False
        for j in range(len(key)):
            if alphabet[i] == key[j]:
                bool_buff = True
                break
        if bool_buff == False:
            new_alphabet.append(alphabet[i])
    mtx_abt_j = []
    counter = 0
    for j in range(6):
        mtx_abt_i = []
        for i in range(6):
            mtx_abt_i.append(new_alphabet[counter])
            counter = counter + 1
        mtx_abt_j.append(mtx_abt_i)
    # проверка на одинаковые биграммы
    for i in range(len(text) - 1):
        if text[i] == text[i + 1]:
            if text[i] != 'я':
                text = text[:i + 1] + 'я' + text[i + 1:]
            else:
                text = text[:i + 1] + 'ю' + text[i + 1:]
    # проверка на четную длину текста
    if len(text) % 2 == 1:
        text = text + "я"
    enc_text = ""
    for t in range(0, len(text), 2):
        flag = True
        for j_1 in range(6):
            if flag == False:
                break
            for i_1 in range(6):
                if flag == False:
                    break
                if mtx_abt_j[j_1][i_1] == text[t]:
                    for j_2 in range(6):
                        if flag == False:
                            break
                        for i_2 in range(6):
                            if mtx_abt_j[j_2][i_2] == text[t+1]:
                                if j_1 != j_2 and i_1 != i_2:
                                    enc_text = enc_text + \
                                        mtx_abt_j[j_1][i_2] + \
                                        mtx_abt_j[j_2][i_1]
                                elif j_1 == j_2 and i_1 != i_2:
                                    enc_text = enc_text + \
                                        mtx_abt_j[j_1][(i_1+1) % 6] + \
                                        mtx_abt_j[j_2][(i_2+1) % 6]
```

```
                                elif j_1 != j_2 and i_1 == i_2:
                                    enc_text = enc_text + \
                                        mtx_abt_j[(j_1+1) % 5][i_1] + \
                                        mtx_abt_j[(j_2+1) % 5][i_2]
                                elif j_1 == j_2 and i_1 == i_2:
                                    enc_text = enc_text + \
                                        mtx_abt_j[j_1][i_1] + \
                                        mtx_abt_j[j_1][i_1]
                                flag = False
                                break
    return enc_text

# функция расшифрования
def playfair_decode(clearText, key):
    text = clearText
    new_alphabet = []
    for i in range(len(key)):
        new_alphabet.append(key[i])
    for i in range(len(alphabet)):
        bool_buff = False
        for j in range(len(key)):
            if alphabet[i] == key[j]:
                bool_buff = True
                break
        if bool_buff == False:
            new_alphabet.append(alphabet[i])
    mtx_abt_j = []
    counter = 0
    for j in range(6):
        mtx_abt_i = []
        for i in range(6):
            mtx_abt_i.append(new_alphabet[counter])
            counter = counter + 1
        mtx_abt_j.append(mtx_abt_i)
    if len(text) % 2 == 1:
        text = text + "я"
    enc_text = ""
    for t in range(0, len(text), 2):
        flag = True
        for j_1 in range(6):
            if flag == False:
                break
            for i_1 in range(6):
                if flag == False:
                    break
                if mtx_abt_j[j_1][i_1] == text[t]:
                    for j_2 in range(6):
                        if flag == False:
                            break
                        for i_2 in range(6):
                            if mtx_abt_j[j_2][i_2] == text[t+1]:
                                if j_1 != j_2 and i_1 != i_2:
                                    enc_text = enc_text + \
```

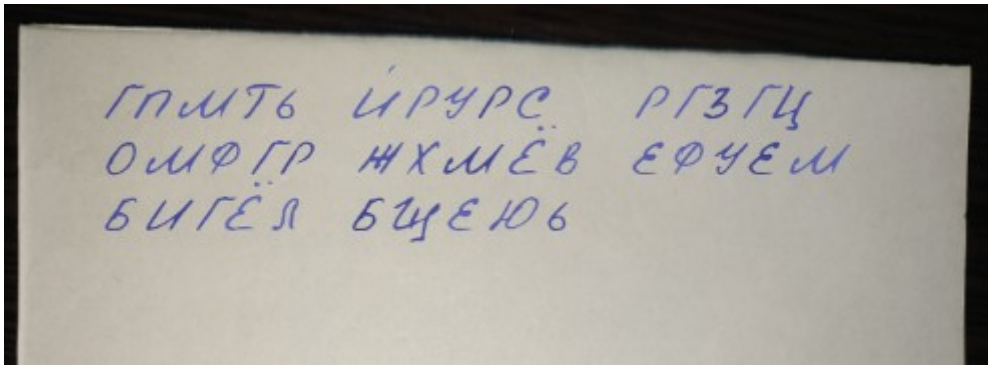
```
                                        mtx_abt_j[j_1][i_2] + \
                                        mtx_abt_j[j_2][i_1]
                                elif j_1 == j_2 and i_1 != i_2:
                                    enc_text = enc_text + \
                                        mtx_abt_j[j_1][(i_1-1) % 6] + \
                                        mtx_abt_j[j_2][(i_2-1) % 6]
                                elif j_1 != j_2 and i_1 == i_2:
                                    enc_text = enc_text + \
                                        mtx_abt_j[(j_1-1) % 5][i_1] + \
                                        mtx_abt_j[(j_2-1) % 5][i_2]
                                elif j_1 == j_2 and i_1 == i_2:
                                    enc_text = enc_text + \
                                        mtx_abt_j[j_1][i_1] + \
                                        mtx_abt_j[j_1][i_1]
                                flag = False
                                break
    return enc_text

#вывод результатов работы программы
print(f'''
Шифр Плейфера:
Ключ: {key}
Таблица : {table}
КОРОТКИЙ ТЕКСТ:
Зашифрованный текст:
{playfair_encode(input_for_cipher_short(), key)}

Расшифрованный текст:
{output_from_decrypted(playfair_decode(playfair_encode(
    input_for_cipher_short(), key), key))}

ДЛИННЫЙ ТЕКСТ:
Зашифрованный текст:
{playfair_encode(input_for_cipher_long(), key)}

Расшифрованный текст:
{output_from_decrypted(playfair_decode(playfair_encode(
    input_for_cipher_long(), key), key))}
''')
```

Тестирование:

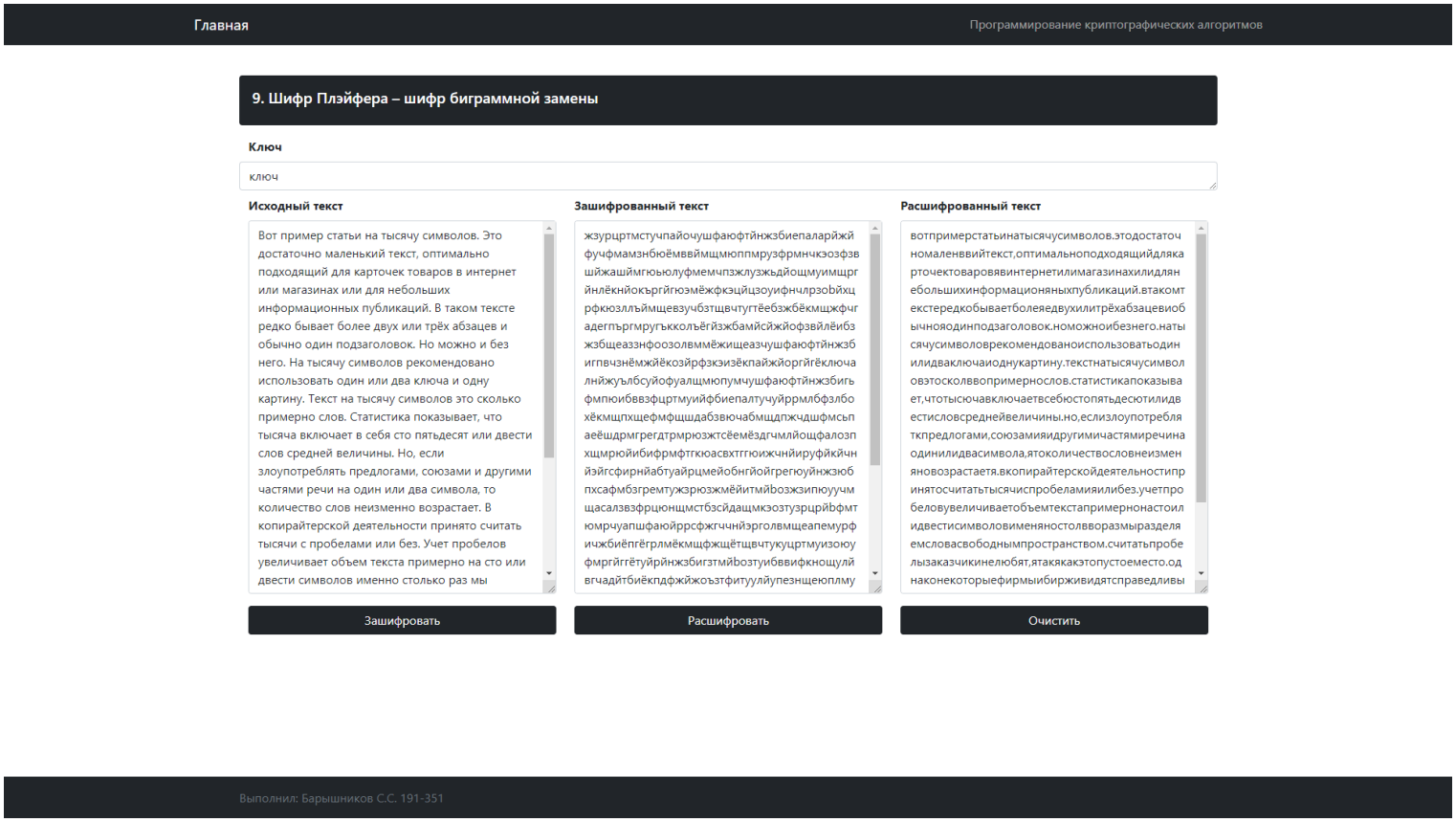
```
/bin/python3
/root/mospolytech-education-crypt-dev-2021-1/lab03_9_playfair.py

Шифр Плейфера:
Ключ: ключ
Таблица:
['к', 'л', 'ю', 'ч', 'а', 'б'],
['в', 'г', 'д', 'е', 'ё', 'ж'],
['з', 'и', 'й', 'м', 'н', 'о'],
['п', 'р', 'с', 'т', 'у', 'ф'],
['х', 'ц', 'ш', 'щ', 'ъ', 'ы'],
```

```
['ъ', 'э', 'я', 'а', 'b', 'c']
КОРОТКИЙ ТЕКСТ:
Зашифрованный текст:
гпмтьйрурсргзгцомфгржхмёвефуембигёлбщеюь

Расшифрованный текст:
время,приливыиотливынеждутчеловека.я

ДЛИННЫЙ ТЕКСТ:
Зашифрованный текст:
жзурцртмстучпайочушфаюфтйнжзбиепаларйжйфучфмамзнбюёмввймщмюппмрузфрмнчкэо
зфзвшйжашймгюьюлуфмемчпзжлузжгзмугтмёрмгрнчёлийуёцзгргюbйжчибяхзцйоыфтилъ
йзннхцрфкюзллъймщевзучбзтщвчтугтёебзжбёкмщжфчгёеёпцзгрусвъбкнкщггзфжщбозй
жйофзвйлёибзжзбщеаззнфоозолвммёжищеазчушфаюфтйнжзбигпвчзнёмжйёкозйрфзкэиз
ёкпайжйоргйгёключалнйжуълбсуйофуалщмюпумчушфаюфтйнжзбигьфмпюибввзфцртмуий
фбиепалтучуйррмлбфзлбохёкмщпхщефмфщшдабзвючабмщдпжчдшфмсьпаеёшдрмгрегдтрм
рюзжтсёемёздгчмлйощфалозпхщмрюйибифрмфттгкюасвхттгюижчнйируфйкйчниигсрёйнм
люусанйтглмуёйжйоргйгёкрйзеибкнруфмбзгремтужзрюзжмёйитмннзжзиултучёттализв
зфрцюнщмстбзсйдащмкэозтузрцрйbфмтюмрчуапшфаюйррсфжгччниигрчжмпалтамщрсфжг
чзжпёгчмлзгчёфмаымтщмюпучрсйнгтозуётузйгрегдтрмрйзеибзжйнёмозтуибввифкнощ
улйвгчадйтбиёкпдфжйжоъзтфитуулйупезнщеюплмучпарсфжгчхоблкнлмлзмёючюспмруу
чккбларзффтфммтдтфмщебзёйблзовчмфифщжротицолофгзгйгаструлгёгюзгщотукёмряп
фмйнйфпанкфщшдатрйзеибзжтрфичжюбнйпхутлмучьсйфчгёймгёкёощоггмтёмфмзчбадтп
еёмозжижзтрцрасйэщеюпижюбрйщмпяпхщемрчуяпгрумшощмюпфчвмёейоижзффирфпюкнру
ойчпзожчсёмщщеазфжибяхйотуёпуъёольёмкнчушфаюпнуёбзжквмрсфжгчзжщеюь

Расшифрованный текст:
вотпримерстатьинатысячусимволов.этодостаточномаленввийтекст,оптимальнопо
дходящийдлякарточектоваровв интернетилимагазинахилидлянебольшихинформацио
нныхпубликаций.втакомтекстередкобываетболеедвухилитрёхабзацевиобычноодинп
одзаголовок.номожноибезнего.натысячусимволоврекомендованоиспользоватьоди
нилидваключаиоднукартину.текстнатысячусимволовэтосколввопримерослов.ста
тистикапоказывает,чтотысючавключаетвсебюстопятьдесютилидвестисловсредней
величины.но,еслизлоупотреблятькпредлогами,союзамиидругимичастямиречинаоди
нилидвасимвола,токоличествословнеизменновозрастает.вкопирайтерскойдеятел
ьностипринятосчитатьтысячиспробеламиилибез.учетпробеловувеличиваетобъемт
екстапримернонастоилидвестисимволовименностолвворазмыразделяемсловасвобо
днымпространством.считатьпробелызаказчикинелюбят,таккакэтопустоеместо.од
наконекоторыефирмыибиржividятсправедливымставитьстоимостьзатысючусимволо
вспробелами,считаяпоследниеважнымгементомкачественноговосприятия.соглас
итесь,читатьслитныйтекстбезединогопропуска,никтонебудет.нобольшинствунуж
наценазатысячузнаковбезпробелов.я
```

Карточка:



Интерфейс:

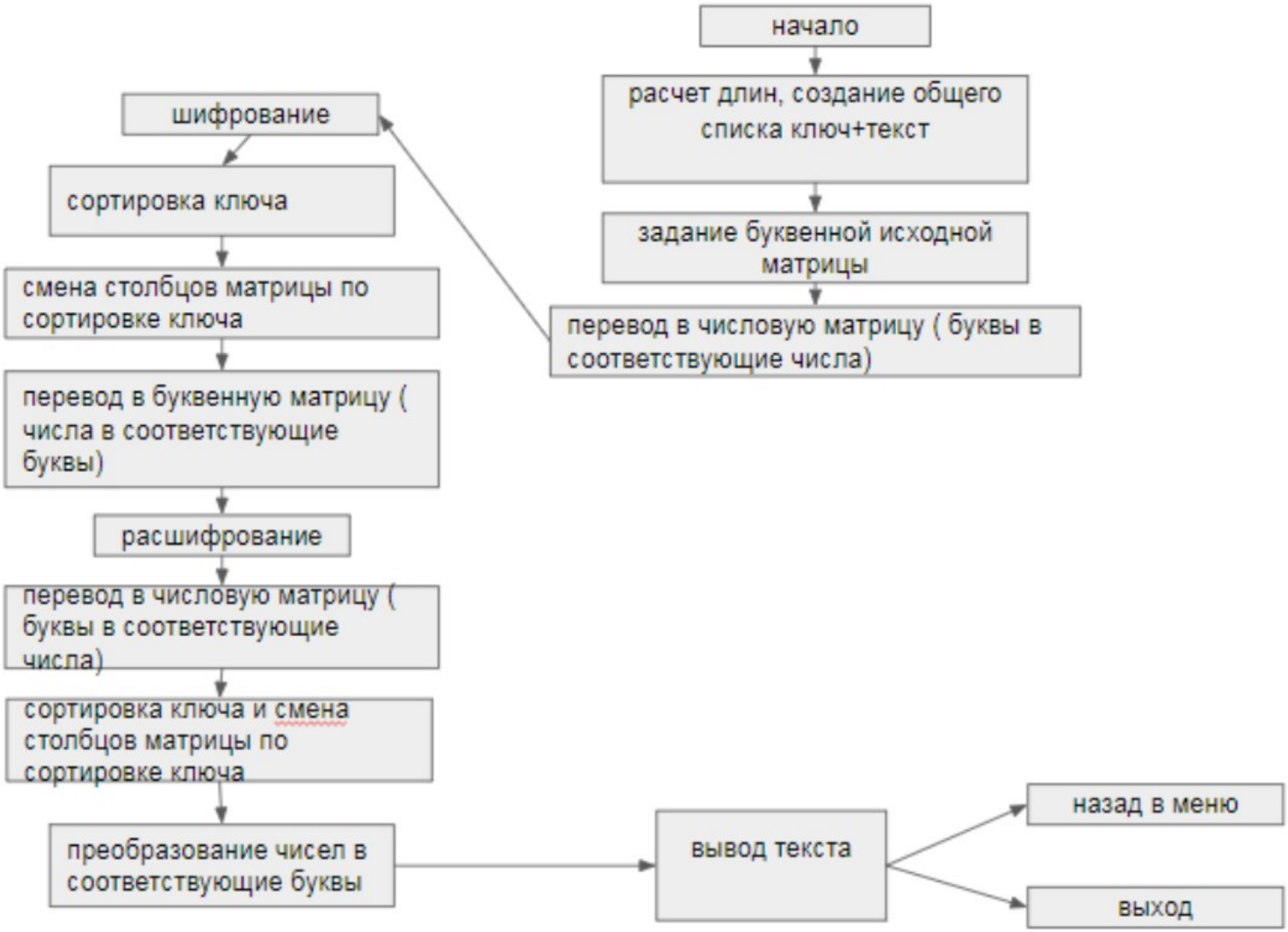


D: ШИФРЫ ПЕРЕСТАНОВКИ

10.Шифр вертикальной перестановки

Широкое распространение получила разновидность маршрутной перестановки — вертикальная перестановка. В этом шифре также используется прямоугольная таблица, в которую сообщение записывается по строкам слева направо. Выписывается шифрограмма по вертикалям, при этом столбцы выбираются в порядке, определяемом ключом.

Блок-схема:



Код программы:

```
# импорт компонентов, необходимых для работы программы
from base import alphabet, input_for_cipher_short,
input_for_cipher_long, output_from_decrypted
import math

# установка ключа
key = 'ключ'

# функция шифрования
def transposition_encode(msg, key):
```

```
    cipher = ""

    k_indx = 0

    msg_len = float(len(msg))
    msg_lst = list(msg)
    key_lst = sorted(list(key))

    col = len(key)

    row = int(math.ceil(msg_len / col))

    matrix = [msg_lst[i: i + col] for i in range(0, len(msg_lst), col)]

    for _ in range(col):
        curr_idx = key.index(key_lst[k_indx])
        cipher += ''.join([row[curr_idx] for row in matrix])
        k_indx += 1

    return cipher

# функция расшифрования
def transposition_decode(cipher, key):
    msg = ""

    k_indx = 0

    msg_indx = 0
    msg_len = float(len(cipher))
    msg_lst = list(cipher)

    col = len(key)

    row = int(math.ceil(msg_len / col))

    key_lst = sorted(list(key))

    dec_cipher = []
    for _ in range(row):
        dec_cipher += [[None] * col]

    for _ in range(col):
        curr_idx = key.index(key_lst[k_indx])

        for j in range(row):
            dec_cipher[j][curr_idx] = msg_lst[msg_indx]
            msg_indx += 1
        k_indx += 1

    msg = ''.join(sum(dec_cipher, []))

    return msg
```

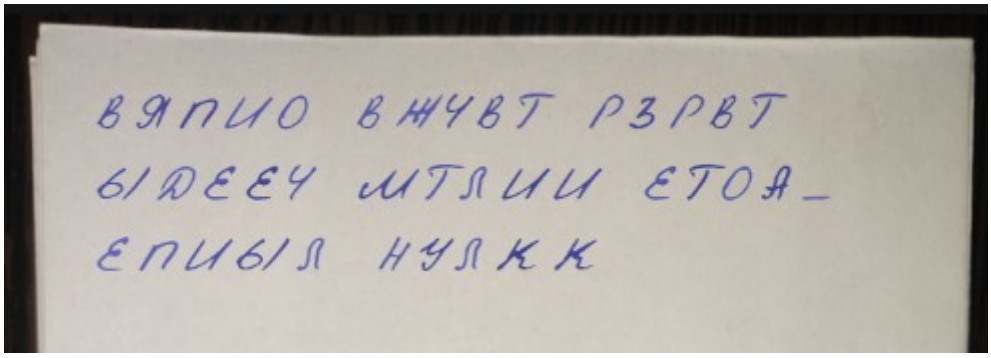
```
#вывод результатов работы программы
print(f'''
Шифр вертикальной перестановки:
Ключ: {key}
КОРОТКИЙ ТЕКСТ:
Зашифрованный текст:
{transposition_encode(input_for_cipher_short(), key)}

Расшифрованный текст:
{output_from_decrypted(transposition_decode(transposition_encode(
    input_for_cipher_short(), key), key))}

ДЛИННЫЙ ТЕКСТ:
Зашифрованный текст:
{transposition_encode(input_for_cipher_long(), key)}

Расшифрованный текст:
{output_from_decrypted(transposition_decode(transposition_encode(
    input_for_cipher_long(), key), key))}
''')
```

Тестирование:

```
/bin/python3
/root/mospolytech-education-crypt-dev-2021-1/lab04_10_transposition.py

Шифр вертикальной перестановки:
Ключ: ключ
КОРОТКИЙ ТЕКСТ:
Зашифрованный текст:
вяпиовжчвтрзрвтыдеечмтлииетоаепиылнулкк

Расшифрованный текст:
время,приливыиотливынеждутчеловека.

ДЛИННЫЙ ТЕКСТ:
Зашифрованный текст:
врртаяилчотчаьттомндядачооннлгнляоифанпииккеекквбехтацонидоокоинтаяилееви
лвоивюокичкаяилтоомовсикквзтсвчвяпдтдтоейиынтиуряроисадиамчоивматлсснмор
ачоарйтнинсаыиоалзупеуиабттинслесоиноомзялсомснокпеачнбпкэуеткаеофырисв
иситозссосбмттодвыетатнвриклтзиьтттеорспкеекоивжеаянвпетоисьтчмокданлкез
паохщлреввтеиааинлхоцнукйвокроаодирбебонзлкнжбечтчмокнасьадлачданкстчмоо
лпесттсаааповякассяеививдвчтоезпетегзомрмсиидлавзоитлееваекпйсдеопячтссбм
итчрлвчеъеамотисилмолрыдеовдптсмсарлкиеятктсмоокриижднпевттосаяилпеисасн
аммочвооиясаептснебдгокттбтнлнуннтчабрлчпеанссоттсомнйстиьодйкотририаиил
бшнмохлцчаттдытеуихаичдогвчмозонссормоооьндлиуттенссоэккиноктиоытчыаютб
оьяислрелнкплотлплмтзигчяеандилтоеовзнзтткреояьтиоттчрлиектбвлвомсрраиви
ввнткзалсабыоавчиьбззиюзакпосчннтемивталмвсмыуввоапипеенлнкснопттчгиьчти
йсзнпузиндчбштуцзсзозбв_тмтиыуввэотоеикптлпоияткаветмзхдеьириыбаттмсебел
влёзвыопаотонегкыувводнпзтииканруттыуввсьррлчатпзеттчлееттслесснеичзслобь
дапюиуитрниисопкчвоиноствиткелсртиьяпеибчеооеитекпенодтмоесьаремвонрртгч
тоыаклттаотетдоорбиярдыаьиттчморлзчялижэемеегсятоссталыкеиопаноутоъснаа
ыукеоок
```

```
Расшифрованный текст:
вотпримерстатьинатысячусимволов.этодостаточномаленькийтекст,оптимальнопо
дходящийдлякарточектовароввинтернетилимагазинахилидлянебольшихинформацио
нныхпубликаций.втакомтекстередкобываетболеедвухилитрёхабзацевиобычноодинп
одзаголовок.номожноибезнего.натысячусимволоврекомендованоиспользоватьоди
нилидваключаиоднукартину.текстнатысячусимволовэтосколькопримернослов.ста
тистикапоказывает,чтотысячавключаетвсебястопятьдесятилидвестисловсредней
величины.но,еслизлоупотреблятьпредлогами,союзамиидругимичастямиречинаоди
нилидвасимвола,токоличествословнеизменновозрастает.вкопирайтерскойдеятел
ьностипринятосчитатьтысячиспробеламиилибез.учетпробеловувеличиваетобъемт
екстапримернонастоилидвестисимволовименностолькоразмыразделяемсловасвобо
днымпространством.считатьпробелызаказчикинелюбят,таккакэтопустоеместо.од
наконекоторыефирмыибиржividятсправедливымставитьстоимостьзатысячусимволо
вспробелами,считаяпоследниеважнымэлементомкачественноговосприятия.соглас
итесь,читатьслитныйтекстбезединогопропуска,никтонебудет.нобольшинствунуж
наценазатысячузнаковбезпробелов.
```

Карточка:



Интерфейс:

10. Вертикальная перестановка

Ключ

ключ

Исходный текст

Вот пример статьи на тысячу символов. Это достаточно маленький текст, оптимально подходящий для карточек товаров в интернет или магазинах или для небольших информационных публикаций. В таком тексте редко бывает более двух или трёх абзацев и обычно один подзаголовок. Но можно и без него. На тысячу символов рекомендовано использовать один или два ключа и одну картину. Текст на тысячу символов это сколько примерно слов. Статистика показывает, что тысяча включает в себя сто пятьдесят или двести слов средней величины. Но, если злоупотреблять предлогами, союзами и другими частями речи на один или два символа, то количество слов неизменно возрастает. В копирайтерской деятельности принято считать тысячи с пробелами или без. Учет пробелов увеличивает объем текста примерно на сто или двести символов именно столько раз мы

Зашифрованный текст

врртаяилчотчаьттомндядачоонннлгнляоифанпиик
кеекквбехтацонидоокоинтаяилеевилвоивюокичк
аяилтоомовсикквзтсвчвяпдтдтоейиынтиуряроис
адиамчоивматлссннморачоарйтнинсаыиоалзупеу
иабттинслесоиноомзялсомснокпеачнбпкэуеткае
офырисвиситозссобмттодвыетатнвриклтзиьттте
орспкеекоивжеаянвпетоисьтчмокданлкезпаохщ
лреввтеиааинлхоцнукйвокроаодирбебонзлкнжб
ечтчмокнасьадлачданкстчмоолпесттсааапоякасс
яеивидвчтоезпетегзомрмсиидлавзоитлееваекп
йсдеопячтссбмитчрлвчеъеамотисилмолрыдеовд
птсмсарлкиеятктсмооккриижднпевттосаяилпеисас
наммочвооиясаептснебдгокттбтнлнуннтчабрлчп
еанссоттсомнйстиьодйкотририаиилбшнмохлцча
ттдытеуихаичдогвчмозонссормоооьндлиуттенс
соэккиноктиоытчыаютбоьяислрелнкплотлплмтзи
гчяеандилтоеовзнзтткреояьтиоттчрлиектбвлвом
срраививвнтказлсабыоавчиьбззиюзакпосчннтем
ивталмвсмыуввоапипеенлнкснопттчгиьчтийсзнп
узиндчбштуцзсзозбв_тмтиыуввэотоеикптлпоиятк

Расшифрованный текст

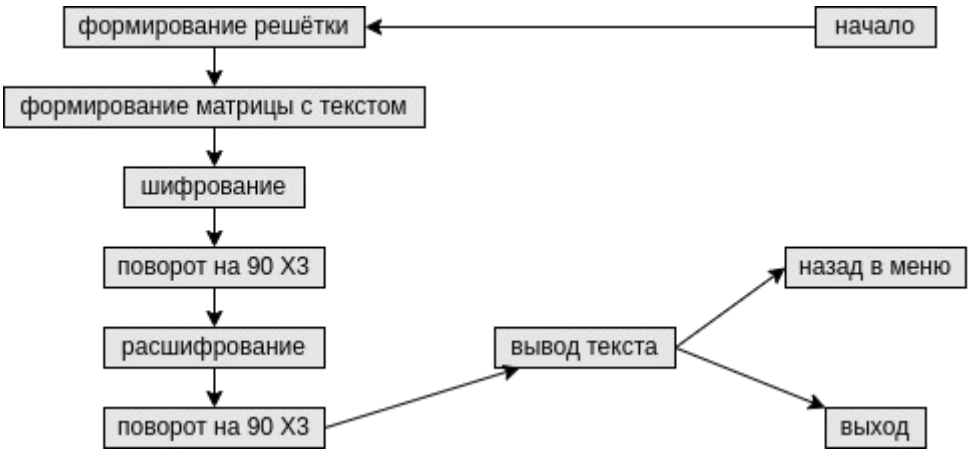
вотпримерстатьинатысячусимволов.этодостаточ
номаленькийтекст,оптимальноподходящийдляка
рточектовароввинтернетилимагазинахилидляне
большихинформационныхпубликаций.втакомтек
стередкобываетболеедвухилитрёхабзацевиобыч
ноодинподзаголовок.номожноибезнего.натысяч
усимволоврекомендованоиспользоватьодинили
двакючаиоднукартину.текстнатысячусимволовэ
тосколькопримернослов.статистикапоказывает,ч
тотысячавключаетвсебястопятьдесятилидвестисл
овсреднейвеличины.но,еслизлоупотреблятьпред
логами,союзамиидругимичастямиречинаодинил
идвасимвола,токоличествословнеизменновозра
стает.вкопирайтерскойдеятельностипринятосчит
атьтысячиспробеламиилибез.учетпробеловувел
ичиваетобъемтекстапримернонастоилидвестиси
мволовименностолькоразмыразделяемсловасво
боднымпространством.считатьпробелызаказчик
инелюбят,таккакэтопустоеместо.однаконекотор
ыефирмыибиржividятсправедливымставитьсто

Зашифровать | Расшифровать | Очистить

11.Решетка Кардано

Решётка Кардано — исторически первая известная шифровальная решётка, трафарет, применявшийся для шифрования и дешифрования, выполненный в форме прямоугольной (чаще всего — квадратной) таблицы-карточки, часть ячеек которых вырезана, и через которые наносился шифротекст. Пустые поля текста заполнялись другим текстом для маскировки сообщений под обычные послания — таким образом, применение решётки является одной из форм стеганографии.

Блок-схема:



Код программы:

```
# импорт компонентов, необходимых для работы программы
from base import alphabet, input_for_cipher_short,
input_for_cipher_long, output_from_decrypted

# объявление класса
class Cardan(object):
    # функция инициализации класса
    def __init__(self, size, spaces):
        self.size = int(size)
        str1 = ''
        for i in range(len(spaces)):
            str1 = str1 + str(spaces[i][0]) + str(spaces[i][1])
        self.spaces = str1
        matrix_spaces = []
        i = 0
        cont = 0
        while i < self.size*self.size//4:
            t = int(self.spaces[cont]), int(self.spaces[cont + 1])
            cont = cont + 2
            i = i+1
            matrix_spaces.append(t)
        self.spaces = matrix_spaces

    # функция шифрования
    def code(self, message):
        offset = 0
        cipher_text = ""
```

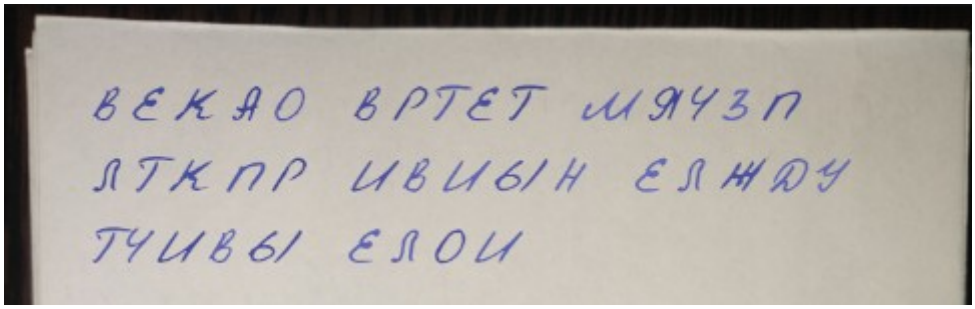
```
        matrix = []
        for i in range(self.size*2-1):
            matrix.append([])
            for j in range(self.size):
                matrix[i].append(None)
        whitesneeded = self.size*self.size - \
            len(message) % (self.size*self.size)
        if (len(message) % (self.size*self.size) != 0):
            for h in range(whitesneeded):
                message = message + ' '
        while offset < len(message):
            self.spaces.sort()
            for i in range(int(self.size*self.size//4)):
                xy = self.spaces[i]
                x = xy[0]
                y = xy[1]
                matrix[x][y] = message[offset]
                offset = offset + 1
            if (offset % (self.size*self.size)) == 0:
                for i in range(self.size):
                    for j in range(self.size):
                        try:
                            cipher_text = cipher_text + matrix[i][j]
                        except:
                            pass
            for i in range(self.size*self.size//4):
                x = (self.size-1)-self.spaces[i][1]
                y = self.spaces[i][0]
                self.spaces[i] = x, y
        return cipher_text

    # функция расшифрования
    def decode(self, message, size):
        uncipher_text = ""
        offset = 0
        matrix = []
        for i in range(self.size*2-1):
            matrix.append([])
            for j in range(self.size):
                matrix[i].append(None)
        whitesneeded = self.size*self.size - \
            len(message) % (self.size*self.size)
        if (len(message) % (self.size*self.size) != 0):
            for h in range(whitesneeded):
                message = message + ' '
        offsetmsg = len(message) - 1
        while offset < len(message):
            if (offset % (self.size*self.size)) == 0:
                for i in reversed(list(range(self.size))):
                    for j in reversed(list(range(self.size))):
                        matrix[i][j] = message[offsetmsg]
                        offsetmsg = offsetmsg - 1
            for i in reversed(list(range(self.size*self.size//4))):
```

```
                x = self.spaces[i][1]
                y = (self.size-1)-self.spaces[i][0]
                self.spaces[i] = x, y
            self.spaces.sort(reverse=True)
            for i in range(self.size*self.size//4):
                xy = self.spaces[i]
                x = xy[0]
                y = xy[1]
                uncipher_text = matrix[x][y] + uncipher_text
                offset = offset + 1

        return uncipher_text

# установка ключа
gaps = [(7, 7), (6, 0), (5, 0), (4, 0), (7, 1), (1, 1), (1, 2), (4, 1),
        (7, 2), (2, 1), (2, 5), (2, 3), (7, 3), (3, 1), (3, 2), (3, 4)]
r = Cardan(8, gaps)

texto = input_for_cipher_short()

n = len(texto)
encoded = r.code(texto)
decoded = r.decode(encoded, n)

gaps2 = [(7, 7), (6, 0), (5, 0), (4, 0), (7, 1), (1, 1), (1, 2), (4, 1),
         (7, 2), (2, 1), (2, 5), (2, 3), (7, 3), (3, 1), (3, 2), (3, 4)]
r2 = Cardan(8, gaps)

texto_long = input_for_cipher_long()

n = len(texto_long)
encoded_long = r2.code(texto_long)
decoded_long = r2.decode(encoded_long, n)

#вывод результатов работы программы
print(f'''
Решетка Кардано:
Ключ: {gaps}
КОРОТКИЙ ТЕКСТ:
Зашифрованный текст:
{encoded.replace(' ', '')}

Расшифрованный текст:
{output_from_decrypted(decoded)}

ДЛИННЫЙ ТЕКСТ:
Зашифрованный текст:
{encoded_long}

Расшифрованный текст:
{output_from_decrypted(decoded_long)}
''')
```

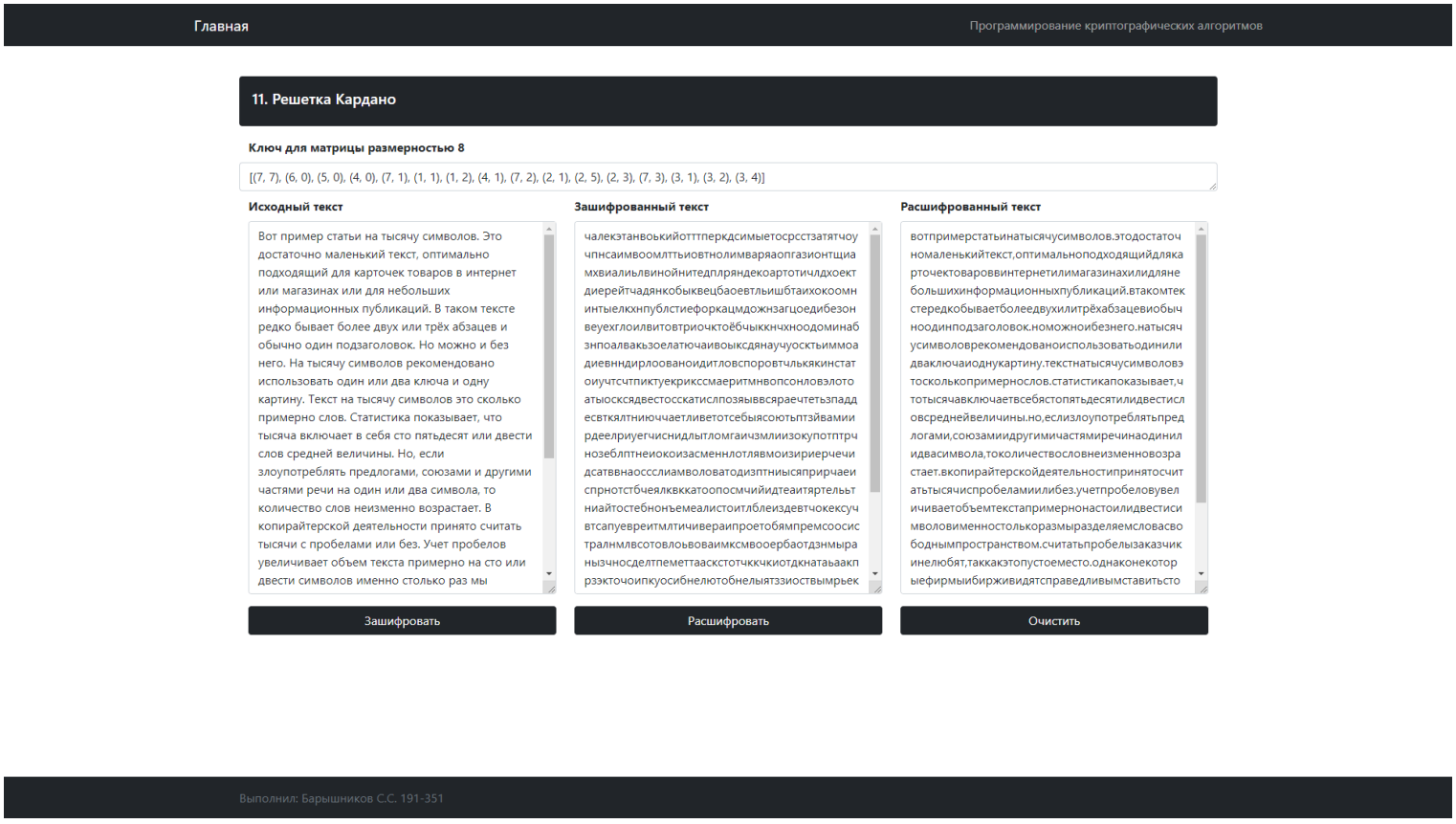
Тестирование:

```
/bin/python3
/root/mospolytech-education-crypt-dev-2021-1/lab04_11_cardan.py

Решетка Кардано:
Ключ: [(7, 7), (6, 0), (5, 0), (4, 0), (7, 1), (1, 1), (1, 2), (4, 1),
(7, 2), (2, 1), (2, 5), (2, 3), (7, 3), (3, 1), (3, 2), (3, 4)]
КОРОТКИЙ ТЕКСТ:
Зашифрованный текст:
векаоврртетмячзплткпривиынелждутчивыелои

Расшифрованный текст:
время,приливыиотливынеждутчеловека.

ДЛИННЫЙ ТЕКСТ:
Зашифрованный текст:
чалекэтанвоькийотттперкдсимыетосрсстзатятчоучпнсаимвоомлттьиовтнолимваря
аопгазионтщиамхвиалиьлвинойнитедплряндекоартотичлдхоектдиерейтчадянкобык
вецбаоевтльишбтаихокоомнинтыелкхнпублстиеформкацмдожнзагцоедибезонвеуехгл
оилвитовтриочктоёбчыккнчхноодоминабзнпоалвакьзоелатючаивоыксдянаучуосктьи
ммоадиевнндирлооваиоидитловспоровтчлькякинстатоиучтсчтпиктуекрикссмаерит
мнвопсонловэлотоатыосксядвестосскатислпозяыввсяраечтетьзпаддесвткялтниюч
чаетливетотсебыясоютьптзйвамиирдеелриуегчиснидлытломгаичзмлиизокупотптрч
нозеблптнеиокоизасменнлотлявмоизириерчечидсатввнаоссслиамволоватодизптни
ысяприрчаеиспрнотстбчеялквккатоопосмчийидтеаитяртелььтниайтостебнонъемеа
листоитлблеиздевтчокексучвтсапуевреитмлтичивераипроетобямпремсоосистралн
млвсотовлоьвоваимксмвооербаотдзнмыранызчносделтпеметтаасктотчккчкиотдкн
атаьаакпрзэкточоипкуосибнелютобнелыятззиоствымрьекзатыссожтяочтурыиесавф
ивиитьирдсятмотмспраимввыибедливачеажнтсолтвенынсвосгмопрчовэлбеиеометл
анятсопаослемкдпмизниептогоныйтприроПутсяеткиаезятсчпксксьттбезопзтендчг
итатиньиласслииевбеназоктпробаеолнлеотвбуъдтысетшяччуитнзснкатчвунукож
кнонацб

Расшифрованный текст:
вотпримерстатьинатысячусимволов.этодостаточномаленькийтекст,оптимальнопо
дходящийдлякарточектоваровввинтернетилимагазинахилидлянебольшихинформацио
нныхпубликаций.втакомтекстередкобываетболеедвухилитрёхабзацевиобычноодинп
одзаголовок.номожноибезнего.натысячусимволоврекомендованоиспользоватьоди
нилидваключаиоднукартину.текстнатысячусимволовэтосколькопримернослов.ста
тистикапоказывает,чтотысячавключаетвсебястопятьдесятилидвестисловсредней
величины.но,еслизлоупотреблятьпредлогами,союзамиидругимичастямиречинаоди
нилидвасимвола,токоличествословнеизменновозрастает.вкопирайтерскойдеятел
ьностипринятосчитатьтысячиспробеламиилибез.учетпробеловувеличиваетобъемт
екстапримернонастоилидвестисимволовименностолькоразмыразделяемсловасвобо
днымпространством.считатьпробелызаказчикинелюбят,таккакэтопустоеместо.од
наконекоторыефирмыибирживидятсправедливымставитьстоимостьзатысячусимволо
вспробелами,считаяпоследниеважнымэлементомкачественноговосприятия.соглас
итесь,читатьслитныйтекстбезединогопропуска,никтонебудет.нобольшинствунуж
наценазатысячузнаковбезпробелов.
```

Карточка:


ВЕКАО ВРТЕТ ЖЯЧЗП
ЯТКПР ИВИБІН ЕЯЖДУ
ТЧИВБІ ЕЯОИ

Интерфейс:


Главная
Программирование криптографических алгоритмов
11. Решетка Кардано
Ключ для матрицы размерностью 8
[(7, 7), (6, 0), (5, 0), (4, 0), (7, 1), (1, 1), (1, 2), (4, 1), (7, 2), (2, 1), (2, 5), (2, 3), (7, 3), (3, 1), (3, 2), (3, 4)]
Исходный текст
Вот пример статьи на тысячу символов. Это достаточно маленький текст, оптимально подходящий для карточек товаров в интернет или магазинах или для небольших информационных публикаций. В таком тексте редко бывает более двух или трёх абзацев и обычно один подзаголовок. Но можно и без него. На тысячу символов рекомендовано использовать один или два ключа и одну картину. Текст на тысячу символов это сколько примерно слов. Статистика показывает, что тысяча включает в себя сто пятьдесят или двести слов средней величины. Но, если злоупотреблять предлогами, союзами и другими частями речи на один или два символа, то количество слов неизменно возрастает. В копирайтерской деятельности принято считать тысячи с пробелами или без. Учет пробелов увеличивает объем текста примерно на сто или двести символов именно столько раз мы
Зашифрованный текст
чалекэтанвоькийотттперкдсимыетосрсстзатятчоучпнсаимвоомлттьиовтнолимваряаопгазионтщиамхвиалиьлвинойнитедплряндекоартотичлдхоектдиерейтчадянкобыквецбаоевтльшбтаихокоомнинтыелкхнпублстиефоркацмдожнзагцоедибезонвеуехглоилвитовтриочктоёбчыккнчхноодоминабзнпоалвакьзоелатючаивоыксдянаучуосктьиммоадиевнндирлооваиоидитловспоровтчлькякинстатоиучтсчтпиктуекрикссмаеритмнвопсонловэлотоатыосксядвестосскатислпозяыввсяраечтетьзпаддесвткялтниоччаетливетотсебыясоютьптзйвамиирдеелриуегчиснидлытломгаичзмлиизокупотптрчнозеблптнеиокоизасменнлотлявмоизириерчечидсатввнаоссслиамволоватодизптниысяприрчаеиспрнотстбчеялквккатоопосмчийидтеаитяртельътниайтостебнонъемеалистоитлбleиздевтчокексучвтсапуевреитмлтичивераипроетобямпремсоосисстралнмлвсотовлоьвоваимксмвооербаотдзнмыранызчносделтпеметтааскстотчккчкиотдкнатаьаакпрзэкточоипкуосибнелютобнелыятззиоствымрьекРасшифрованный текст
вотпримерстатьинатысячусимволов.этодостаточноманенькийтекст,оптимальноподходящийдлякарточектоваровввинтернетилимагазинахилидляне большихинформационныхпубликаций.втакомтекстередкобываетболеедвухилитрёхабзацевиобычноодинподзаголовок.моможноибезнего.натысячусимволоврекомендованоиспользоватьодинилидваключаиоднукартину.текстнатысячусимволовэтосколькопримернослов.статистикапоказывает,чтотысячавключаетвсебястопятьдесятилидвестисловсреднейвеличины.но,еслизлоупотреблятьпредлогами,союзамиидругимичастямиречинаодинилидвасимвола,токоличествословнеизменновозрастает.вкопирайтерскойдеятельностипринятосчитатьтысячиспробеламиилибез.учетпробеловувеличиваетобъемтекстапримернонастоилидвестисимволовименностолькоразмыразделяемсловасвободнымпространством.считатьпробелызаказчикинелюбят,таккакэтопустоеместо.однаконекоторыефирмыибиржividятсправедливымставитьсто
Зашифровать
Расшифровать
Очистить
Выполнил: Барышников С.С. 191-351

Е: ШИФРЫ ГАММИРОВАНИЯ

13.Одноразовый блокнот К.Шеннона

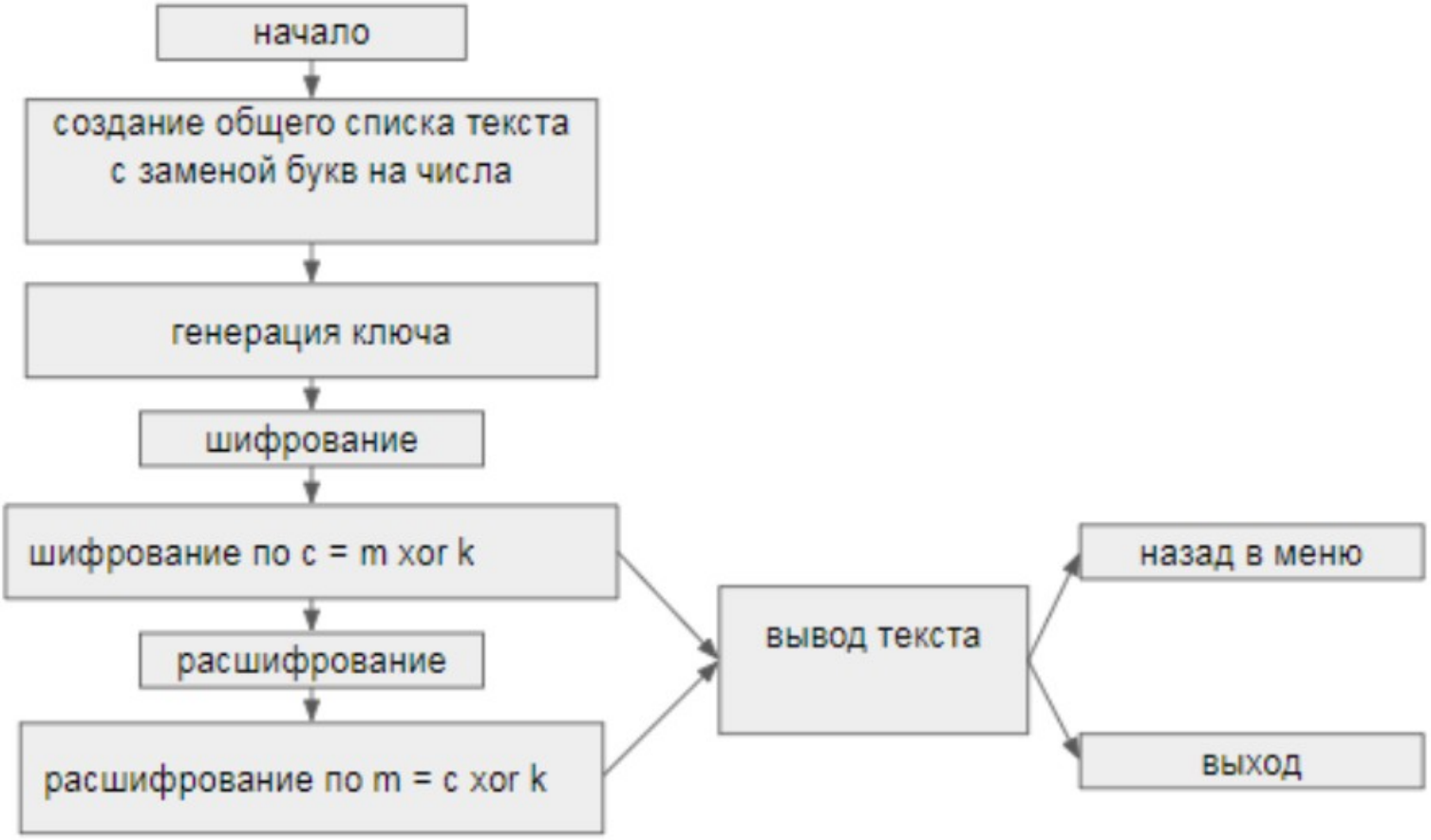
Популярность поточных шифров можно связывать с работой Клода Шеннона, посвященной анализу одноразовых гамма-блокнотов. Название «одноразовый блокнот» стало общепринятым в годы Второй мировой войны, когда для шифрования широко использовались бумажные блокноты.

Одноразовый блокнот использует длинную шифрующую последовательность, которая состоит из случайно выбираемых бит или наборов бит (символов). Шифрующая последовательность побитно или посимвольно накладывается на открытый текст, имеет ту же самую длину, что и открытое сообщение, и может использоваться только один раз (о чем свидетельствует само название шифрсистемы); ясно, что при таком способе шифрования требуется огромное количество шифрующей гаммы.

Открытый текст сообщения ш записывают как последовательность бит или символов m = momi...mn_i, а двоичную или символьную шифрующую последовательность к той же самой длины - как k = koki...k„_|.

Шифртекст c = c0cl...cn.i определяется соотношением Cj = mi Шк, при 0

Блок-схема:



Код программы:

```
# импорт компонентов, необходимых для работы программы
```

```
import random
from base import alphabet, input_for_cipher_short, input_for_cipher_long,
output_from_decrypted

alphabet = alphabet.replace(' ', '')
alphabet_lower = {}
i = 0
while i < (len(alphabet)):
    alphabet_lower.update({alphabet[i]: i})
    i += 1

# фунция получения ключа
def get_key(d, value):
    for k, v in d.items():
        if v == value:
            return k


# функция шифрования
def shenon_encode(msg):
    msg_list = list(msg)
    msg_list_len = len(msg_list)
    msg_code_bin_list = list()
    for i in range(len(msg_list)):
        msg_code_bin_list.append(alphabet_lower.get(msg_list[i]))

    key_list = list()
    for i in range(msg_list_len):
        key_list.append(random.randint(0, 32))

    cipher_list = list()
    for i in range(msg_list_len):
        m = int(msg_code_bin_list[i])
        k = int(key_list[i])
        cipher_list.append(int(bin(m ^ k), base=2))
    return cipher_list, key_list


# функция расшифрования
def shenon_decode(msg, key_list):
    decipher_list = list()
    msg_list_len = len(msg)
    for i in range(msg_list_len):
        c = int(msg[i])
        k = int(key_list[i])
        decipher_list.append(int(bin(c ^ k), base=2))
    deciphered_str = ""
    for i in range(len(decipher_list)):
        deciphered_str += get_key(alphabet_lower, decipher_list[i])
    return deciphered_str


short_encoded = shenon_encode(input_for_cipher_short())
```

```
short_decoded = shenon_decode(short_encoded[0], short_encoded[1])

long_encoded = shenon_encode(input_for_cipher_long())
long_decoded = shenon_decode(long_encoded[0], long_encoded[1])

#вывод результатов работы программы
print(f'''
Одноразовый блокнот:
КОРОТКИЙ ТЕКСТ:
Зашифрованный текст:
{short_encoded[0]}
Ключ:
{short_encoded[1]}

Расшифрованный текст:
{output_from_decrypted(short_decoded)}

ДЛИННЫЙ ТЕКСТ:
Зашифрованный текст:
{long_encoded[0]}
Ключ:
{long_encoded[1]}

Расшифрованный текст:
{output_from_decrypted(long_decoded)}
''')
```

Тестирование:

```
/bin/python3 /root/mospolytech-education-crypt-dev-2021-1/lab05_13_shenon.py

Одноразовый блокнот:
КОРОТКИЙ ТЕКСТ:
Зашифрованный текст:
[20, 24, 3, 1, 54, 21, 5, 27, 10, 6, 19, 22, 12, 23, 30, 22, 12, 20, 31, 13, 30, 6, 17,
14, 21, 0, 0, 4, 20, 10, 7, 10, 0, 3, 15, 9, 26, 31, 26]
Ключ:
[22, 9, 6, 12, 22, 29, 21, 8, 26, 23, 26, 26, 5, 21, 2, 31, 3, 7, 19, 4, 28, 26, 31,
11, 18, 4, 20, 23, 12, 15, 11, 5, 2, 6, 4, 9, 9, 7, 17]

Расшифрованный текст:
время,приливыиотливынеждутчеловека.

ДЛИННЫЙ ТЕКСТ:
Зашифрованный текст:
[12, 47, 30, 29, 1, 28, 12, 13, 17, 28, 14, 8, 8, 13, 23, 25, 7, 14, 25, 7, 49, 56, 27,
11, 13, 16, 10, 24, 9, 14, 3, 26, 22, 6, 9, 6, 7, 0, 6, 5, 7, 31, 7, 12, 12, 11, 1, 15,
32, 25, 24, 4, 31, 17, 7, 9, 2, 8, 23, 9, 17, 17, 48, 4, 3, 19, 1, 10, 1, 4, 27, 9, 25,
23, 15, 17, 29, 12, 13, 3, 47, 21, 22, 12, 19, 19, 58, 10, 8, 12, 11, 14, 31, 31, 31,
9, 6, 18, 1, 49, 3, 10, 27, 3, 19, 21, 26, 16, 13, 2, 5, 21, 15, 26, 5, 5, 5, 32, 29,
28, 9, 14, 25, 3, 22, 15, 6, 14, 34, 14, 7, 15, 18, 0, 29, 1, 27, 14, 12, 46, 16, 27,
2, 4, 27, 16, 16, 10, 12, 0, 21, 1, 28, 24, 17, 1, 8, 27, 19, 13, 11, 16, 20, 9, 1, 34,
23, 19, 25, 7, 20, 11, 1, 14, 23, 9, 10, 21, 4, 10, 29, 6, 3, 6, 6, 9, 23, 0, 4, 0, 14,
```

```
14, 9, 12, 15, 1, 20, 14, 27, 5, 20, 0, 13, 13, 25, 27, 1, 7, 7, 2, 15, 6, 7, 27, 10,
23, 2, 9, 31, 21, 22, 25, 22, 23, 36, 23, 31, 10, 18, 8, 19, 28, 21, 8, 4, 23, 1, 22,
29, 31, 28, 27, 31, 22, 21, 3, 23, 30, 7, 11, 0, 18, 31, 20, 4, 20, 25, 27, 19, 8, 45,
26, 9, 14, 7, 16, 20, 15, 23, 26, 11, 16, 29, 23, 7, 24, 8, 4, 4, 9, 9, 14, 22, 19, 15,
23, 3, 2, 15, 9, 19, 1, 9, 11, 9, 17, 14, 27, 12, 25, 7, 21, 2, 8, 31, 13, 23, 44, 13,
5, 9, 29, 23, 20, 11, 25, 2, 5, 2, 4, 26, 23, 20, 27, 9, 27, 30, 24, 14, 7, 17, 23, 4,
0, 13, 17, 33, 22, 10, 20, 9, 13, 24, 16, 27, 1, 3, 11, 21, 9, 17, 20, 17, 20, 19, 26,
21, 24, 1, 17, 0, 9, 22, 9, 19, 4, 44, 22, 25, 28, 26, 28, 5, 29, 12, 3, 12, 8, 22, 18,
5, 7, 14, 8, 27, 16, 12, 15, 29, 30, 6, 20, 17, 29, 15, 14, 18, 24, 7, 22, 28, 38, 15,
28, 6, 16, 4, 6, 21, 23, 9, 22, 9, 1, 18, 26, 52, 22, 19, 21, 25, 37, 15, 28, 31, 8,
30, 36, 19, 17, 14, 12, 26, 5, 4, 30, 6, 0, 5, 25, 0, 14, 25, 27, 1, 36, 14, 15, 6, 11,
31, 13, 17, 1, 31, 27, 28, 7, 12, 22, 8, 28, 17, 22, 16, 25, 25, 28, 21, 29, 13, 0, 20,
17, 13, 23, 23, 37, 15, 8, 58, 4, 8, 9, 27, 7, 5, 19, 21, 13, 21, 25, 24, 19, 4, 2, 26,
25, 12, 11, 24, 26, 8, 19, 11, 11, 31, 35, 1, 9, 26, 22, 2, 25, 26, 60, 24, 2, 49, 15,
22, 27, 14, 3, 29, 12, 7, 13, 7, 31, 2, 12, 22, 13, 14, 17, 5, 6, 31, 21, 21, 14, 14,
25, 4, 7, 30, 25, 27, 10, 8, 16, 7, 14, 6, 29, 29, 17, 2, 31, 15, 12, 14, 19, 17, 31,
21, 25, 16, 10, 26, 19, 9, 31, 22, 15, 32, 18, 1, 27, 3, 0, 17, 5, 47, 10, 11, 16, 4,
29, 4, 4, 17, 21, 10, 14, 10, 10, 13, 50, 15, 15, 2, 19, 15, 10, 18, 7, 29, 48, 13, 15,
13, 42, 7, 1, 13, 7, 0, 27, 4, 0, 16, 13, 17, 4, 34, 1, 12, 7, 6, 8, 19, 13, 10, 9, 22,
25, 10, 23, 12, 23, 30, 29, 40, 10, 16, 2, 16, 2, 7, 13, 31, 27, 15, 16, 11, 0, 19, 4,
10, 31, 26, 16, 24, 56, 2, 18, 8, 0, 30, 24, 5, 11, 0, 6, 31, 19, 10, 31, 8, 23, 5, 24,
26, 6, 30, 11, 13, 25, 16, 17, 13, 6, 30, 21, 15, 7, 1, 9, 2, 15, 12, 14, 0, 10, 6, 31,
5, 29, 26, 5, 30, 12, 28, 4, 18, 8, 1, 13, 24, 24, 7, 8, 21, 27, 25, 17, 18, 2, 27, 16,
3, 4, 12, 22, 45, 15, 0, 10, 14, 29, 34, 31, 12, 22, 31, 21, 26, 2, 16, 16, 11, 15, 0,
6, 23, 26, 25, 2, 2, 28, 16, 3, 24, 2, 25, 17, 26, 10, 26, 16, 21, 25, 27, 4, 18, 21,
23, 6, 0, 18, 30, 6, 24, 17, 1, 16, 25, 22, 20, 15, 30, 37, 29, 4, 21, 54, 6, 18, 3, 8,
6, 12, 23, 27, 15, 27, 4, 51, 29, 30, 7, 14, 31, 5, 0, 18, 18, 29, 31, 5, 27, 19, 21,
30, 4, 4, 21, 31, 31, 46, 3, 16, 18, 4, 12, 25, 5, 22, 28, 7, 11, 9, 29, 23, 29, 22,
11, 12, 10, 7, 24, 29, 35, 7, 13, 10, 12, 12, 17, 17, 1, 14, 17, 22, 19, 21, 20, 16, 0,
20, 17, 51, 13, 30, 10, 24, 2, 25, 20, 21, 22, 17, 13, 23, 13, 25, 29, 41, 25, 20, 26,
23, 25, 12, 20, 1, 26, 34, 11, 16, 20, 29, 20, 17, 44, 5, 21, 13, 27, 7, 17, 50, 21, 0,
13, 31, 53, 31, 27, 21, 19, 26, 24, 17, 41, 5, 30, 21, 8, 14, 16, 45, 3, 22, 37, 17,
21, 20, 25, 12, 45, 26, 14, 14, 1, 50, 18, 21, 22, 5, 18, 24, 17, 24, 13, 28, 21, 10,
16, 6, 52, 1, 30, 35, 17, 23, 2, 25, 1, 20, 19, 9, 20, 7, 18, 26, 3, 9, 11, 31, 11, 22,
26, 7, 11, 15, 16, 31, 20, 8, 11, 20, 11, 5, 0, 27, 24, 1, 9, 30, 22, 23, 14, 7, 41,
28, 31, 16, 30, 21, 10, 28, 19, 10, 14, 25, 10, 2, 19, 21, 27, 0, 11, 26, 26, 27, 30,
12, 21, 4, 10, 0, 10, 25, 13, 11, 20, 13, 31, 19, 27, 15, 29, 22, 8, 21, 31, 21, 11, 9,
0, 15, 10, 7, 4, 17, 14, 16, 24, 9, 22, 3, 55, 17, 3, 6, 46, 23, 21, 14, 18, 18, 1, 4,
18, 25, 25, 0, 16, 7, 8, 19, 51, 25, 30]
Ключ:
[14, 32, 13, 13, 16, 21, 1, 8, 0, 14, 29, 8, 27, 16, 30, 23, 7, 29, 5, 21, 17, 32, 15,
25, 4, 29, 8, 23, 5, 1, 1, 9, 14, 13, 23, 21, 8, 4, 9, 23, 20, 31, 20, 3, 20, 5, 14, 2,
32, 21, 29, 10, 2, 26, 14, 3, 17, 13, 28, 27, 2, 25, 32, 23, 12, 3, 18, 3, 12, 4, 23,
20, 23, 24, 31, 30, 25, 26, 2, 7, 15, 15, 31, 6, 23, 31, 26, 1, 8, 29, 24, 1, 7, 26,
20, 26, 9, 16, 1, 32, 12, 8, 25, 10, 29, 6, 31, 1, 3, 7, 22, 28, 3, 19, 8, 5, 6, 32,
21, 21, 7, 14, 15, 10, 26, 6, 2, 2, 2, 0, 2, 14, 29, 12, 0, 24, 18, 24, 5, 32, 5, 20,
19, 9, 27, 7, 25, 5, 2, 14, 9, 23, 12, 12, 16, 13, 1, 16, 19, 26, 2, 26, 7, 17, 10, 32,
4, 19, 18, 8, 25, 24, 4, 5, 5, 26, 15, 4, 1, 14, 22, 9, 2, 26, 4, 9, 18, 19, 5, 15, 2,
11, 12, 8, 13, 21, 2, 7, 23, 12, 7, 17, 11, 27, 25, 26, 9, 7, 16, 7, 13, 15, 8, 26, 22,
15, 12, 6, 16, 17, 31, 23, 6, 24, 32, 31, 31, 9, 29, 4, 28, 30, 26, 3, 23, 15, 10, 24,
18, 18, 19, 28, 17, 25, 28, 2, 18, 22, 9, 14, 3, 29, 12, 12, 15, 26, 25, 8, 15, 26, 13,
2, 29, 28, 14, 29, 22, 0, 27, 21, 9, 1, 24, 28, 8, 21, 13, 10, 0, 6, 11, 14, 24, 28, 6,
5, 19, 13, 3, 20, 27, 14, 11, 11, 26, 12, 1, 31, 5, 23, 14, 25, 11, 12, 29, 13, 28, 32,
```

```
18, 29, 9, 20, 24, 16, 5, 13, 9, 5, 19, 23, 19, 25, 0, 8, 17, 16, 13, 29, 5, 21, 2, 25,
4, 19, 17, 3, 1, 14, 30, 6, 0, 0, 26, 31, 23, 14, 1, 21, 6, 6, 3, 31, 30, 24, 14, 17,
26, 8, 16, 24, 13, 12, 7, 7, 28, 22, 32, 25, 27, 15, 2, 23, 23, 14, 12, 16, 5, 26, 5,
27, 14, 7, 30, 7, 16, 16, 4, 19, 31, 30, 3, 7, 25, 13, 28, 22, 1, 23, 20, 10, 14, 6,
23, 28, 4, 27, 8, 25, 13, 23, 12, 5, 11, 19, 23, 27, 20, 4, 0, 26, 9, 5, 28, 1, 27, 13,
12, 4, 0, 24, 2, 5, 30, 7, 1, 12, 21, 9, 23, 21, 15, 12, 11, 10, 4, 32, 0, 10, 12, 9,
26, 1, 24, 25, 22, 21, 0, 20, 20, 29, 6, 19, 25, 6, 3, 28, 11, 16, 28, 21, 1, 15, 0, 1,
2, 4, 6, 32, 14, 4, 26, 23, 21, 25, 10, 2, 1, 31, 26, 14, 21, 20, 17, 27, 20, 17, 8,
22, 19, 3, 24, 23, 1, 26, 15, 26, 11, 32, 8, 4, 19, 14, 2, 11, 9, 28, 21, 11, 32, 10,
14, 18, 0, 3, 18, 8, 14, 3, 14, 19, 11, 8, 20, 13, 28, 24, 8, 4, 16, 25, 21, 6, 30, 10,
23, 8, 21, 22, 23, 3, 16, 21, 21, 29, 4, 18, 15, 29, 13, 29, 1, 9, 7, 27, 28, 26, 27,
23, 31, 8, 21, 27, 24, 31, 4, 28, 32, 23, 18, 8, 27, 11, 19, 14, 32, 26, 2, 1, 4, 23,
23, 1, 0, 7, 1, 1, 0, 14, 8, 18, 28, 10, 14, 14, 1, 5, 0, 20, 20, 32, 28, 6, 3, 10, 20,
14, 31, 31, 9, 8, 4, 19, 13, 30, 13, 22, 2, 25, 5, 21, 22, 25, 28, 12, 15, 5, 22, 20,
3, 30, 0, 30, 31, 24, 32, 25, 8, 9, 4, 26, 2, 30, 15, 10, 0, 17, 14, 12, 28, 6, 30, 29,
31, 28, 17, 32, 11, 16, 8, 5, 13, 23, 4, 16, 5, 11, 12, 22, 1, 13, 27, 23, 21, 9, 19,
11, 27, 26, 3, 22, 30, 17, 31, 21, 17, 28, 3, 14, 5, 11, 7, 29, 31, 7, 18, 3, 11, 29,
10, 17, 21, 7, 23, 1, 25, 10, 28, 7, 19, 30, 23, 20, 26, 3, 26, 10, 25, 25, 31, 30, 10,
16, 11, 0, 9, 26, 13, 10, 13, 24, 2, 18, 32, 31, 30, 20, 16, 20, 21, 6, 30, 12, 6, 31,
17, 9, 5, 9, 8, 2, 12, 14, 3, 1, 23, 15, 10, 9, 17, 24, 2, 25, 6, 25, 8, 25, 2, 4, 24,
7, 5, 30, 2, 14, 24, 26, 1, 24, 1, 31, 31, 6, 16, 32, 17, 27, 20, 22, 21, 26, 19, 27,
21, 12, 28, 16, 15, 16, 26, 32, 18, 14, 19, 28, 12, 10, 5, 31, 23, 15, 12, 10, 8, 11,
30, 17, 0, 10, 21, 20, 16, 32, 6, 27, 29, 23, 3, 8, 25, 19, 9, 14, 26, 4, 1, 30, 28,
31, 26, 11, 3, 5, 17, 25, 3, 20, 31, 26, 29, 12, 19, 20, 5, 2, 24, 20, 15, 24, 6, 3, 0,
22, 24, 32, 16, 12, 25, 23, 11, 20, 27, 7, 5, 12, 5, 23, 30, 5, 15, 9, 1, 0, 8, 30, 20,
14, 27, 13, 21, 32, 25, 0, 5, 18, 21, 20, 32, 5, 24, 4, 19, 23, 2, 32, 13, 9, 30, 31,
21, 15, 20, 7, 31, 31, 28, 31, 32, 0, 28, 21, 15, 0, 12, 32, 29, 26, 32, 28, 16, 26,
10, 3, 32, 17, 14, 22, 4, 32, 1, 23, 19, 11, 28, 23, 18, 23, 15, 19, 7, 26, 1, 15, 20,
18, 23, 3, 2, 15, 9, 11, 14, 23, 31, 9, 6, 14, 1, 31, 17, 20, 3, 15, 24, 14, 19, 20,
11, 28, 13, 13, 24, 1, 24, 26, 23, 15, 19, 30, 19, 19, 26, 31, 19, 31, 11, 3, 32, 18,
16, 19, 17, 5, 27, 19, 3, 30, 28, 18, 10, 10, 3, 6, 21, 9, 0, 9, 21, 21, 27, 13, 1, 0,
15, 19, 25, 1, 6, 5, 27, 12, 16, 31, 6, 22, 20, 24, 26, 6, 29, 1, 5, 29, 7, 1, 10, 16,
1, 31, 14, 24, 24, 26, 10, 17, 23, 9, 23, 14, 32, 23, 30, 1, 16, 19, 4, 12, 2, 8, 22,
1, 21, 11, 7, 17, 32, 1, 21]

Расшифрованный текст:
вотпримерстатьинатысячусимволов.этодостаточномаленькийтекст,оптимальноподходящийдлякарт
очектовароввинтернетилимагазинахилидлянебольшихинформационныхпубликаций.втакомтекстеред
кобываетболеедвухилитрёхабзацевиобычноодинподзаголовок.номожноибезнего.натысячусимволов
рекомендованоиспользоватьодинилидваключаиоднукартину.текстнатысячусимволовэтосколькопри
мерноослов.статистикапоказывает,чтотысячавключаетвсебястопятьдесятилидвестисловсреднейве
личины.но,еслизлоупотреблятьпредлогами,союзамиидругимичастямиречинаодинилидвасимвола,то
количествословнеизменновозрастает.вкопирайтерскойдеятельностипринятосчитатьтысячиспробе
ламиилибез.учетпробеловувеличиваетобъемтекстапримернонастоилидвестисимволовименностольк
оразмыразделяемсловасвободнымпространством.считатьпробелызаказчикинелюбят,таккакэтопуст
оеместо.однаконекоторыефирмыибиржividятсправедливымставитьстоимостьзатысячусимволовспро
белами,считаяпоследниеважнымэлементомкачественноговосприятия.согласитесь,читатьслитныйт
екстбезединогопропуска,никтонебудет.нобольшинствунужнаценазатысячузнаковбезпробелов.
```

Интерфейс:

13. Одноразовый блокнот К.Шеннона

Ключ (появляется после шифрования)

[1, 3, 9, 25, 19, 24, 28, 14, 30, 3, 30, 16, 11, 25, 22, 31, 20, 17, 31, 5, 7, 10, 29, 0, 0, 10, 31, 0, 22, 4, 29, 24, 31, 15, 17, 9, 21, 20, 20, 7, 6, 4, 10, 17, 21, 24, 9, 30, 5, 31, 5, 28, 7, 29, 8, 26,

Исходный текст

Вот пример статьи на тысячу символов. Это достаточно маленький текст, оптимально подходящий для карточек товаров в интернет или магазинах или для небольших информационных публикаций. В таком тексте редко бывает более двух или трёх абзацев и обычно один подзаголовок. Но можно и без него. На тысячу символов рекомендовано использовать один или два ключа и одну картину. Текст на тысячу символов это сколько примерно слов. Статистика показывает, что тысяча включает в себя сто пятьдесят или двести слов средней величины. Но, если злоупотреблять предлогами, союзами и другими частями речи на один или два символа, то количество слов неизменно возрастает. В копирайтерской деятельности принято считать тысячи с пробелами или без. Учет пробелов увеличивает объем текста примерно на сто или двести символов именно столько раз мы

Зашифрованный текст

[3, 12, 26, 9, 2, 17, 17, 11, 15, 17, 13, 16, 24, 4, 31, 17, 20, 2, 3, 23, 39, 18, 9, 18, 9, 7, 29, 15, 26, 11, 31, 11, 7, 4, 15, 26, 26, 16, 27, 21, 21, 4, 25, 30, 13, 22, 6, 19, 5, 19, 0, 18, 26, 22, 1, 16, 1, 7, 27, 28, 17, 40, 12, 7, 17, 31, 17, 28, 8, 26, 2, 14, 7, 18, 7, 11, 23, 22, 24, 5, 42, 23, 12, 16, 25, 14, 59, 1, 5, 18, 2, 4, 8, 37, 3, 5, 28, 18, 3, 12, 15, 20, 7, 28, 22, 27, 31, 18, 23, 16, 14, 15, 25, 10, 3, 25, 19, 2, 24, 16, 13, 7, 27, 30, 0, 41, 24, 17, 32, 29, 6, 11, 23, 27, 19, 24, 29, 22, 16, 10, 28, 4, 8, 2, 23, 22, 24, 20, 3, 21, 22, 7, 15, 14, 3, 10, 14, 16, 0, 26, 11, 15, 21, 4, 16, 23, 4, 28, 4, 30, 17, 4, 30, 1, 5, 22, 13, 28, 15, 21, 29, 25, 21, 22, 27, 27, 37, 2, 14, 11, 15, 6, 12, 15, 3, 0, 29, 11, 26, 17, 0, 27, 31, 2, 10, 18, 9, 17, 3, 14, 12, 2, 27, 33, 30, 13, 5, 9, 17, 29, 10, 16, 11, 13, 3, 5, 8, 22, 14, 44, 8, 25, 23, 3, 5, 31, 10, 19, 24, 29, 3, 18, 11, 22, 41, 14, 3, 9, 25, 4, 8, 0, 11, 4, 7, 3, 9, 31, 27, 12, 32, 2, 7, 5, 15, 14, 9, 18, 25, 7, 10, 24, 14, 6, 9, 6, 3, 30, 25, 22, 10, 4, 8, 2, 0, 13, 5, 47, 4, 24, 15, 25, 25, 15, 30, 8, 17, 17, 22, 13, 19, 11, 15, 30, 9, 23, 10, 11, 27, 12, 1, 19, 24, 14, 8, 28, 28, 15, 4, 25,

Расшифрованный текст

вотпримерстатьинатысячусимволов.этодостаточноманенькийтекст,оптимальноподходящийдляка рточектовароввинтернетилимагазинахилидляне большихинформационныхпубликаций.втакомтек стередкобываетболеедвухилитрёхабзацевиобычн оодинподзаголовок.номожноибезнего.натысяч усимволоврекомендованоиспользоватьодинили двакючаиоднукартину.текстнатысячусимволовэ тосколькопримернослов.статистикапоказывает,ч тотысячавключаетвсебястопятьдесятилидвестисл овсреднейвеличины.но,еслизлоупотреблятьпред логами,союзамиидругимичастямиречинаодинил идвасимвола,токоличествословнеизменновозра стает.вкопирайтерскойдеятельностипринятосчит атьтысячиспробеламиилибез.учетпробеловувел ичиваетобъемтекстапримернонастоилидвестиси мволовименностолькоразмыразделяемсловасво боднымпространством.считатьпробелызаказчик инелюбят,таккакэтопустоеместо.однаконекотор ыефирмыибиржividятсправедливымставитьсто

Зашифровать | Расшифровать | Очистить

14.Гаммирование ГОСТ 28147-89

При работе ГОСТ 28147-89 в режиме гаммирования описанным выше образом формируется криптографическая гамма, которая затем побитно складывается по модулю 2 с исходным открытым текстом для получения шифротекста. Шифрование в режиме гаммирования лишено недостатков, присущих режиму простой замены. Так, даже идентичные блоки исходного текста дают разный шифротекст, а для текстов с длиной, не кратной 64 бит, "лишние" биты гаммы отбрасываются. Кроме того, гамма может быть выработана заранее, что соответствует работе шифра в поточном режиме.

Блок-схема:

Код программы:

```
# -*- coding:utf-8 -*-
# импорт компонентов, необходимых для работы программы
import sys
import numpy.random
import itertools
from base import alphabet, input_for_cipher_short,
input_for_cipher_long, output_from_decrypted
import binascii


# объявление класса
class GostCrypt(object):
    def __init__(self, key, sbox):
        self._key = None
        self._subkeys = None
        self.key = key
        self.sbox = sbox

    @staticmethod
    def _bit_length(value):
        return len(bin(value)[2:])

    @property
    def key(self):
        return self._key
```

```
    @key.setter
    def key(self, key):
        self._key = key
        self._subkeys = [(key >> (32 * i)) &
                         0xFFFFFFFF for i in range(8)]

    def _f(self, part, key):
        temp = part ^ key
        output = 0
        for i in range(8):
            output |= ((self.sbox[i][(temp >> (4 * i)) & 0b1111]) << (4 *
i))
        return ((output >> 11) | (output << (32 - 11))) & 0xFFFFFFFF

    def _decrypt_round(self, left_part, right_part, round_key):
        return left_part, right_part ^ self._f(left_part, round_key)

    # функция шифрования
    def encrypt(self, plain_msg):
        def _encrypt_round(left_part, right_part, round_key):
            return right_part, left_part ^ self._f(right_part, round_key)

        left_part = plain_msg >> 32
        right_part = plain_msg & 0xFFFFFFFF
        for i in range(24):
            left_part, right_part = _encrypt_round(
                left_part, right_part, self._subkeys[i % 8])
        for i in range(8):
            left_part, right_part = _encrypt_round(
                left_part, right_part, self._subkeys[7 - i])
        return (left_part << 32) | right_part

    # функция расшифрования
    def decrypt(self, crypted_msg):
        def _decrypt_round(left_part, right_part, round_key):
            return right_part ^ self._f(left_part, round_key), left_part

        left_part = crypted_msg >> 32
        right_part = crypted_msg & 0xFFFFFFFF
        for i in range(8):
            left_part, right_part = _decrypt_round(
                left_part, right_part, self._subkeys[i])
        for i in range(24):
            left_part, right_part = _decrypt_round(
                left_part, right_part, self._subkeys[(7 - i) % 8])
        return (left_part << 32) | right_part


# объявление S-блока
sbox = [numpy.random.permutation(l)
        for l in itertools.repeat(list(range(16)), 8)]
sbox = (
    (4, 10, 9, 2, 13, 8, 0, 14, 6, 11, 1, 12, 7, 15, 5, 3),
```

```
    (14, 11, 4, 12, 6, 13, 15, 10, 2, 3, 8, 1, 0, 7, 5, 9),
    (5, 8, 1, 13, 10, 3, 4, 2, 14, 15, 12, 7, 6, 0, 9, 11),
    (7, 13, 10, 1, 0, 8, 9, 15, 14, 4, 6, 12, 11, 2, 5, 3),
    (6, 12, 7, 1, 5, 15, 13, 8, 4, 10, 9, 14, 0, 3, 11, 2),
    (4, 11, 10, 0, 7, 2, 1, 13, 3, 6, 8, 5, 9, 12, 15, 14),
    (13, 11, 4, 1, 3, 15, 5, 9, 0, 10, 14, 7, 6, 8, 2, 12),
    (1, 15, 13, 0, 5, 7, 10, 4, 9, 2, 3, 14, 6, 11, 8, 12),
)

# установка ключа
key =
183182793879123879127893789123798218793879782387932788723783298329823980
23031

text_short = input_for_cipher_short().encode().hex()
text_short = int(text_short, 16)

gost_short = GostCrypt(key, sbox)

encode_text_short = gost_short.encrypt(text_short)
decode_text_short = gost_short.decrypt(encode_text_short)
decode_text_short = bytes.fromhex(hex(decode_text_short)
[2::]).decode('utf-8')


text_long = input_for_cipher_long().encode().hex()
text_long = int(text_long, 16)

gost_long = GostCrypt(key, sbox)

encode_text_long = gost_long.encrypt(text_long)
decode_text_long = gost_long.decrypt(encode_text_long)
decode_text_long = bytes.fromhex(hex(decode_text_long)[2::]).decode('utf-
8')

#вывод результатов работы программы
print(f'''
Гост 28147-89:
Ключ: {key}
КОРОТКИЙ ТЕКСТ:
Зашифрованный текст:
{encode_text_short}

Расшифрованный текст:
{output_from_decrypted(decode_text_short)}

ДЛИННЫЙ ТЕКСТ:
Зашифрованный текст:
{encode_text_long}

Расшифрованный текст:
{output_from_decrypted(decode_text_long)}
''')
```

Тестирование:

```
/bin/python3
/root/mospolytech-education-crypt-dev-2021-1/lab05_14_gost89.py

Гост 28147-89:
Ключ:
1831827938791238791278937891237982187938797823879327887237832983298239802
23031
КОРОТКИЙ ТЕКСТ:
Зашифрованный текст:
5675402614518369628605669058311409646330599621682397254008495707145036151
6686534633542086281498134144445869433834001779406256238181699835567875396
7671152907458756001251775951863714096309068 2

Расшифрованный текст:
время,приливыиотливынеждутчеловека.

ДЛИННЫЙ ТЕКСТ:
Зашифрованный текст:
2660504179384763560870172300148736432061604479961145151290917749597470334
0594389034389714286970445081539368360502368487388800490982968045229502871
8546978021649941177826880151732274533643864532320796630272450746837347468
4283063682350895571195422576014055891201234951917117574410777206443142305
5027034775088399845189867145831434001642575501399559517052919052602493497
5357719363564132761031905290355995297001142437984799362147596854803459571
5017650775261104984599500375814441113013017875969917042030848511398372774
5509962073921913391810397788675115132457776580708464051846373071577643730
1972347270270738541956306030682103473176355115450382471426481231331613070
6264074393244483028493575194280183695841632846694677441664704015987166764
3738658517814221245915214928246803162017812436951725385369568312910693127
7213053865907702662820017898901782710245581128464419744614772822737082379
5695828448846841904902145096572681358073870106763470339835923246436026070
8890382002875053592506820582408502620331704501617095262512488892027835394
5660411684805903983648279450857063552411750584170039072470197525824827940
6107572222767582208189240451665976152382603851822752929335645908960132784
1046712363123251360022267584062361921103062944008184059512152151700998698
9040967708294011350207395634358154279036795782188750678109140959896732193
6432916690830836157099844286928789002776683458441918340765274704251161726
9389753273012998279245334616877159640347816015296464295681617647172047594
3282516939890220675604110486639041389619497175244677978921983486852096179
0344530438359259087681089419095818947367924242461052662246177551436759868
0609644131641613792985187497438227966239919276204302869135347925244725013
8561606228343378826613603418244138259685282867881546424889824265595654998
7693016474857317967045765805402303929238917825768453421398261020477598060
0009464910092710806727389232375505337101980972052186720533409389553482212
1423383888251650643168732003624170059793831256089491353050782028804944025
9042991842353486749848132500272518620867206880491021944412872541231317285
7315960092225568479643477408050023889229147527678395405300014514788559927
9208235047400922773599948384461446382922212048102735317321789072559748723
7394377717448927872036003024551320639628668092313266559193295747938168888
1713391447550197512421414080155618078837436716254239495738949352604395115
1164131904106002370372557788499423181314 0
```

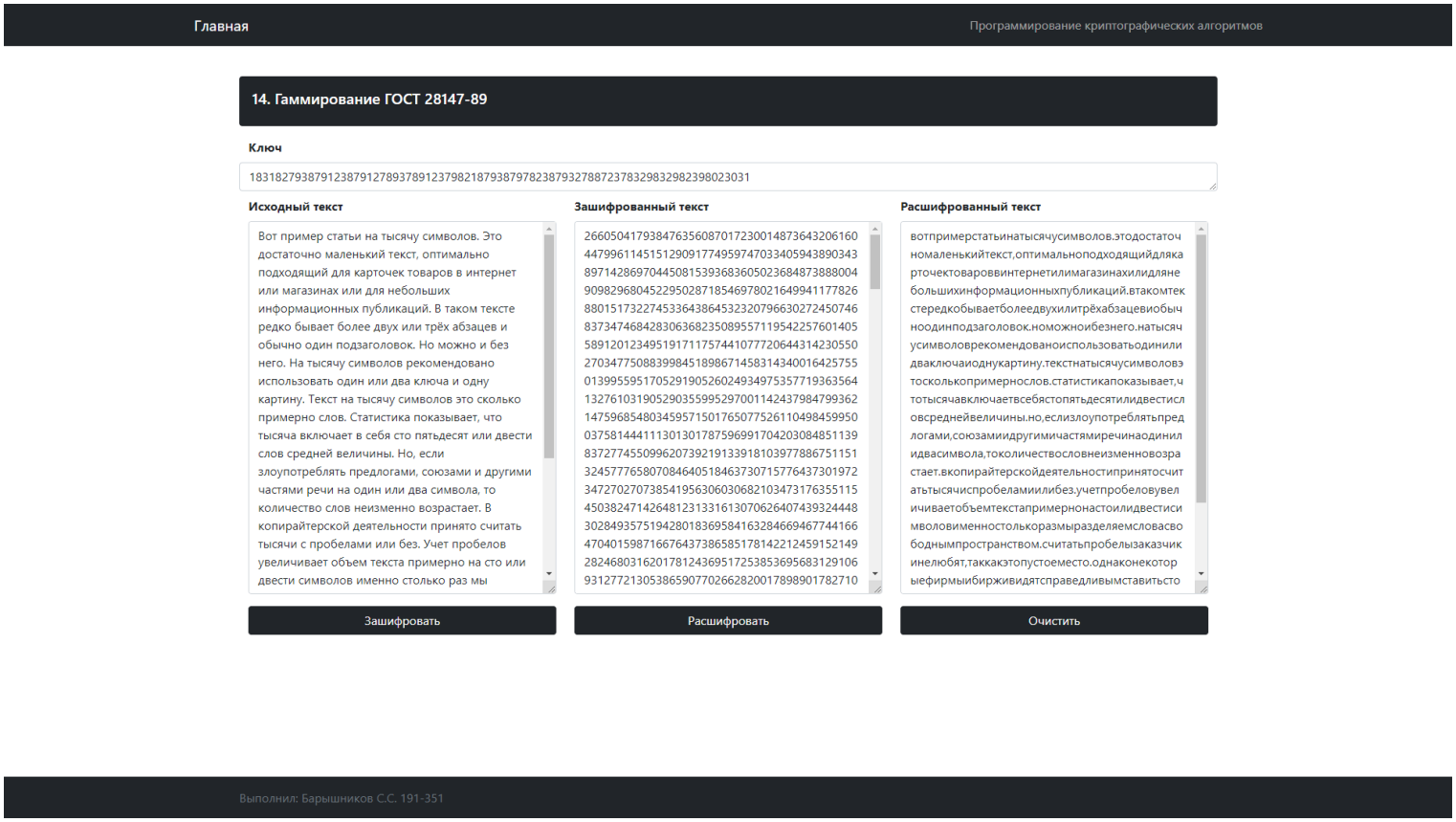
```
048786799371661224581550866678838661153598348870084078780660806173864959
575440976259810399260985123639014703405899712008192601514010680293658982
087391990002313959687075225053283505950476973029918940426894245371802777
391073036436447830978355730100191029987692864338226777977329734416050940
748805043510405396860180398796105350833009786586672656643919674255713516
478226090845673012833291177598368532072347268280649646088861803172128773
291473109506515106659314836040036656092490429237666178198432598615370604
918596863952703155533932809708613849358306640566150863185187925326215754
989333634208585037942116948570784641239076408273805977542386315236522220
735837042084864232903289614347772814746287308830124185850051835267046045
917229478062649585331391527428177596959793939040858892101319105275363913
304553570537217715021676410150446502090474544839536320632177801846664387
486546536087426856664230543677661182546696219642719572277711104465600090
082600855546657041321254047486700934548367850139484608188565474092236157
024928018111515171154353076248616600260838281861827510318136996678149054 6
552935014522113343273944814098061456935761847648355086285555631781660278
333417692540611644909139555372683089914174258373570685999137976226937578
374646893469473239456558064456440375957265656396640816066359824776600968
566072302055241997176140082268654237339173231406859280719003360127669764
491083657763669822633064823655370148189944849907605500443563608149069730
141447001572786353044965071938488320817393982327350914870757149168519557
298244867969723184161812866514205609360545347238801513987968950761463626
793219961349481860117569449510682396573567985224845334032332288473731984
742216366368332972770750786521814163957991317872146879701156367505884840
443242033743649284403534308730028565578448899028468749368165959133099234
922065256529641077966134974017515306130528864541755878618653332828287755
685594822441777128735697886946641091243111681545238116551858244306426392
836846104064731076842543578816818530479152156759120290446042559868977983
946528986507504554880362587909904988045775663409621628751758337985082711
409428127972518695971904771679899146399020169600159539495666103044643078
840967245774105859770442044412377244765493396419787742155086335373263474
448018184154889873746151758484429412090320498815046405341806128814394905
868219934450400886138337839240264255614068324070696593204945639821400860
155559484459548932550840102882387267858402968600171176157376831434777201
707285299833699125037728269831720745214081011576306227271226229939415986
451030780497468026208293099379288210543447284509744576781872462063625033
331957011656270150086434012363118225202497856559102049032543407053333746
197469855990322303022144902536648886278160402357782913621359570037109561
666619129528015842754809484320410643238137797569327685399314897468891322
774686848219935886576012631365200360985488217832469 4
```

Расшифрованный текст:

вотпримерстатьинатысячусимволов.этодостаточномаленькийтекст,оптимальнопо
дходящийдлякарточектовароввинтернетилимагазинахилидлянебольшихинформацио
нныхпубликаций.втакомтекстередкобываетболеедвухилитрёхабзацевиобычноодинп
одзаголовок.номожноибезнего.натысячусимволоврекомендованоиспользоватьоди
нилидваключаиоднукартину.текстнатысячусимволовэтосколькопримернослов.ста
тистикапоказывает,чтотысячавключаетвсебястопятьдесятилидвестисловсредней
величины.но,еслизлоупотреблятьпредлогами,союзамиидругимичастямиречинаоди
нилидвасимвола,токоличествословнеизменновозрастает.вкопирайтерскойдеятел
ьностипринятосчитатьтысячиспробеламиилибез.учетпробеловувеличиваетобъемт
екстапримернонастоилидвестисимволовименностолькоразмыразделяемсловасвобо
днымпространством.считатьпробелызаказчикинелюбят,таккакэтопустоеместо.од

наконекоторыефирмыибиржividятсправедливымставитьстоимостьзатысячусимволовспробелами,считаяпоследниеважнымэлементомкачественноговосприятия.согласитесь,читатьслитныйтекстбезединогопропуска,никтонебудет.нобольшинствунужнаценазатысячузнаковбезпробелов.

Интерфейс:



F: ПОТОЧНЫЕ ШИФРЫ

15.A5 /1

A5 — это поточный алгоритм шифрования, используемый для обеспечения конфиденциальности передаваемых данных между телефоном и базовой станцией в европейской системе мобильной цифровой связи GSM (Groupe Spécial Mobile).

Шифр основан на побитовом сложении по модулю два (булева операция «исключающее или») генерируемой псевдослучайной последовательности и шифруемой информации. В A5 псевдослучайная последовательность реализуется на основе трёх линейных регистров сдвига с обратной связью. Регистры имеют длины 19, 22 и 23 бита соответственно. Сдвигами управляет специальная схема, организующая на каждом шаге смещение как минимум двух регистров, что приводит к их неравномерному движению. Последовательность формируется путём операции «исключающее или» над выходными битами регистров.

Блок-схема:

Код программы:

```
# -*- coding:utf-8 -*-
# импорт компонентов, необходимых для работы программы
from base import alphabet, input_for_cipher_short,
input_for_cipher_long, output_from_decrypted
import re
```

```
import copy
reg_x_length = 19
reg_y_length = 22
reg_z_length = 23

key_one = ""
reg_x = []
reg_y = []
reg_z = []

# функция загрузки регистров
def loading_registers(key):
    i = 0
    while(i < reg_x_length):
        reg_x.insert(i, int(key[i]))
        i = i + 1
    j = 0
    p = reg_x_length
    while(j < reg_y_length):
        reg_y.insert(j, int(key[p]))
        p = p + 1
        j = j + 1
    k = reg_y_length + reg_x_length
    r = 0
    while(r < reg_z_length):
        reg_z.insert(r, int(key[k]))
        k = k + 1
        r = r + 1

# функция установки ключа
def set_key(key):
    if(len(key) == 64 and re.match("^([01])+", key)):
        key_one = key
        loading_registers(key)
        return True
    return False

# фунция получения ключа
def get_key():
    return key_one

# функция перевода текста в бинарный код
def to_binary(plain):
    s = ""
    i = 0
    for i in plain:
        binary = str(' '.join(format(ord(x), 'b') for x in i))
        j = len(binary)
        while(j < 12):
            binary = "0" + binary
            s = s + binary
            j = j + 1
    binary_values = []
```

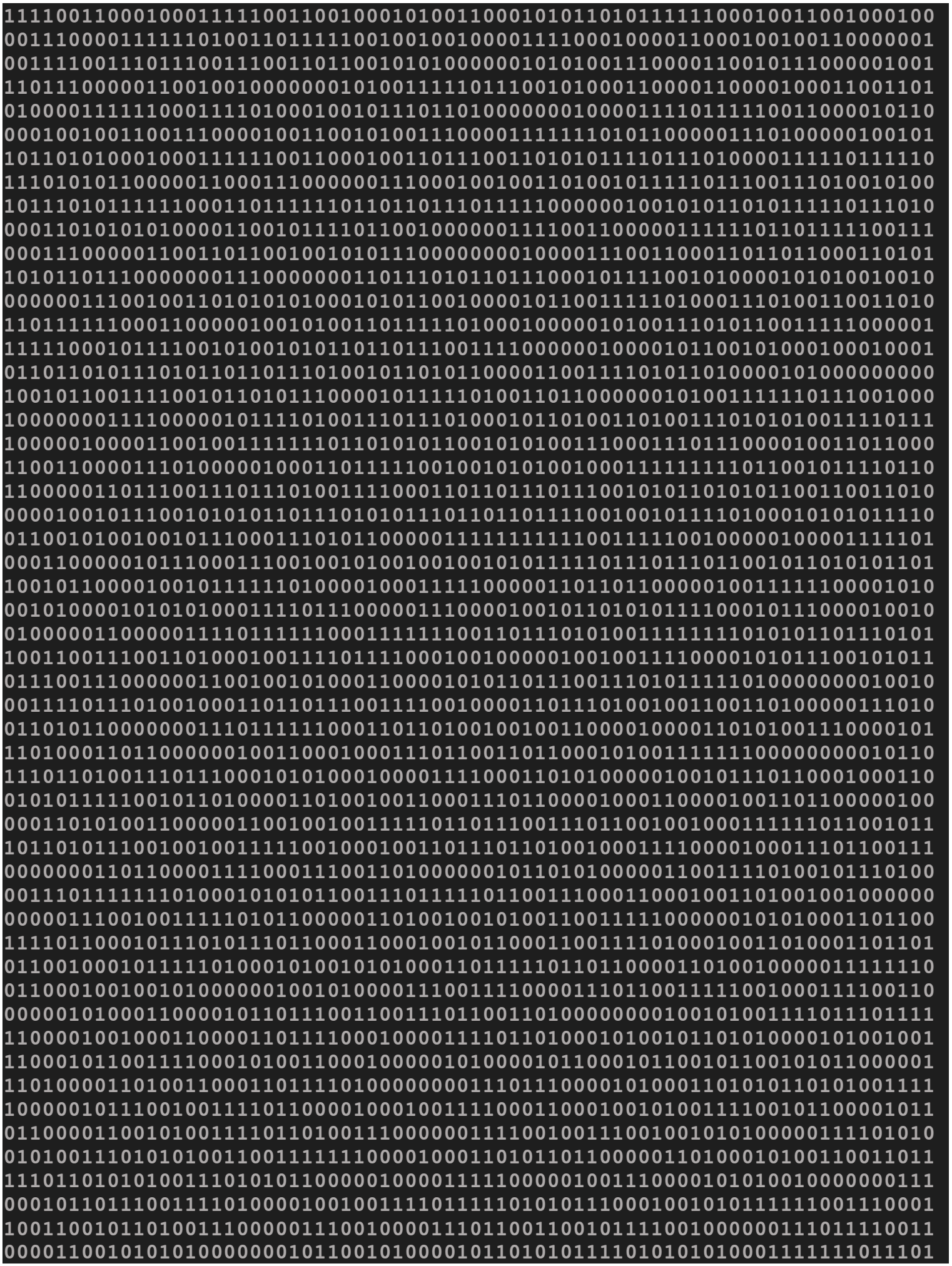
```
    k = 0
    while(k < len(s)):
        binary_values.insert(k, int(s[k]))
        k = k + 1
    return binary_values

def get_majority(x, y, z):
    if(x + y + z > 1):
        return 1
    else:
        return 0

# функция получения потока
def get_keystream(length):
    reg_x_temp = copy.deepcopy(reg_x)
    reg_y_temp = copy.deepcopy(reg_y)
    reg_z_temp = copy.deepcopy(reg_z)
    keystream = []
    i = 0
    while i < length:
        majority = get_majority(reg_x_temp[8], reg_y_temp[10],
reg_z_temp[10])
        if reg_x_temp[8] == majority:
            new = reg_x_temp[13] ^ reg_x_temp[16] ^ reg_x_temp[17] ^
reg_x_temp[18]
            reg_x_temp_two = copy.deepcopy(reg_x_temp)
            j = 1
            while(j < len(reg_x_temp)):
                reg_x_temp[j] = reg_x_temp_two[j-1]
                j = j + 1
            reg_x_temp[0] = new

        if reg_y_temp[10] == majority:
            new_one = reg_y_temp[20] ^ reg_y_temp[21]
            reg_y_temp_two = copy.deepcopy(reg_y_temp)
            k = 1
            while(k < len(reg_y_temp)):
                reg_y_temp[k] = reg_y_temp_two[k-1]
                k = k + 1
            reg_y_temp[0] = new_one

        if reg_z_temp[10] == majority:
            new_two = reg_z_temp[7] ^ reg_z_temp[20] ^ reg_z_temp[21] ^
reg_z_temp[22]
            reg_z_temp_two = copy.deepcopy(reg_z_temp)
            m = 1
            while(m < len(reg_z_temp)):
                reg_z_temp[m] = reg_z_temp_two[m-1]
                m = m + 1
            reg_z_temp[0] = new_two

        keystream.insert(i, reg_x_temp[18] ^ reg_y_temp[21] ^
reg_z_temp[22])
```

```
        i = i + 1
    return keystream

# функция перевода бинарного кода в текст
def convert_binary_to_str(binary):
    s = ""
    length = len(binary) - 12
    i = 0
    while(i <= length):
        s = s + chr(int(binary[i:i+12], 2))
        i = i + 12
    return str(s)

# функция шифрования
def encrypt(plain):
    s = ""
    binary = to_binary(plain)
    keystream = get_keystream(len(binary))
    i = 0
    while(i < len(binary)):
        s = s + str(binary[i] ^ keystream[i])
        i = i + 1
    return s

# функция расшифрования
def decrypt(cipher):
    s = ""
    binary = []
    keystream = get_keystream(len(cipher))
    i = 0
    while(i < len(cipher)):
        binary.insert(i, int(cipher[i]))
        s = s + str(binary[i] ^ keystream[i])
        i = i + 1
    return convert_binary_to_str(str(s))

# функция проверки введенного ключа
def user_input_key():
    tha_key = str(input('Введите 64-bit ключ: '))
    if (len(tha_key) == 64 and re.match("^([01])+", tha_key)):
        return tha_key
    else:
        while(len(tha_key) != 64 and not re.match("^([01])+", tha_key)):
            if (len(tha_key) == 64 and re.match("^([01])+", tha_key)):
                return tha_key
            tha_key = str(input('Введите 64-bit ключ: '))
    return tha_key

# установка ключа
key = '0101001000011010110001110001100100101001000000110111111010110111'
set_key(key)

#вывод результатов работы программы
```

```
print(f'''
A5/1:
Ключ: {key}
КОРОТКИЙ ТЕКСТ:
Зашифрованный текст:
{encrypt(input_for_cipher_short())}

Расшифрованный текст:
{output_from_decrypted(decrypt(encrypt(
    input_for_cipher_short())))}

ДЛИННЫЙ ТЕКСТ:
Зашифрованный текст:
{encrypt(input_for_cipher_long())}

Расшифрованный текст:
{output_from_decrypted(decrypt(encrypt(
    input_for_cipher_long())))}
''')
```

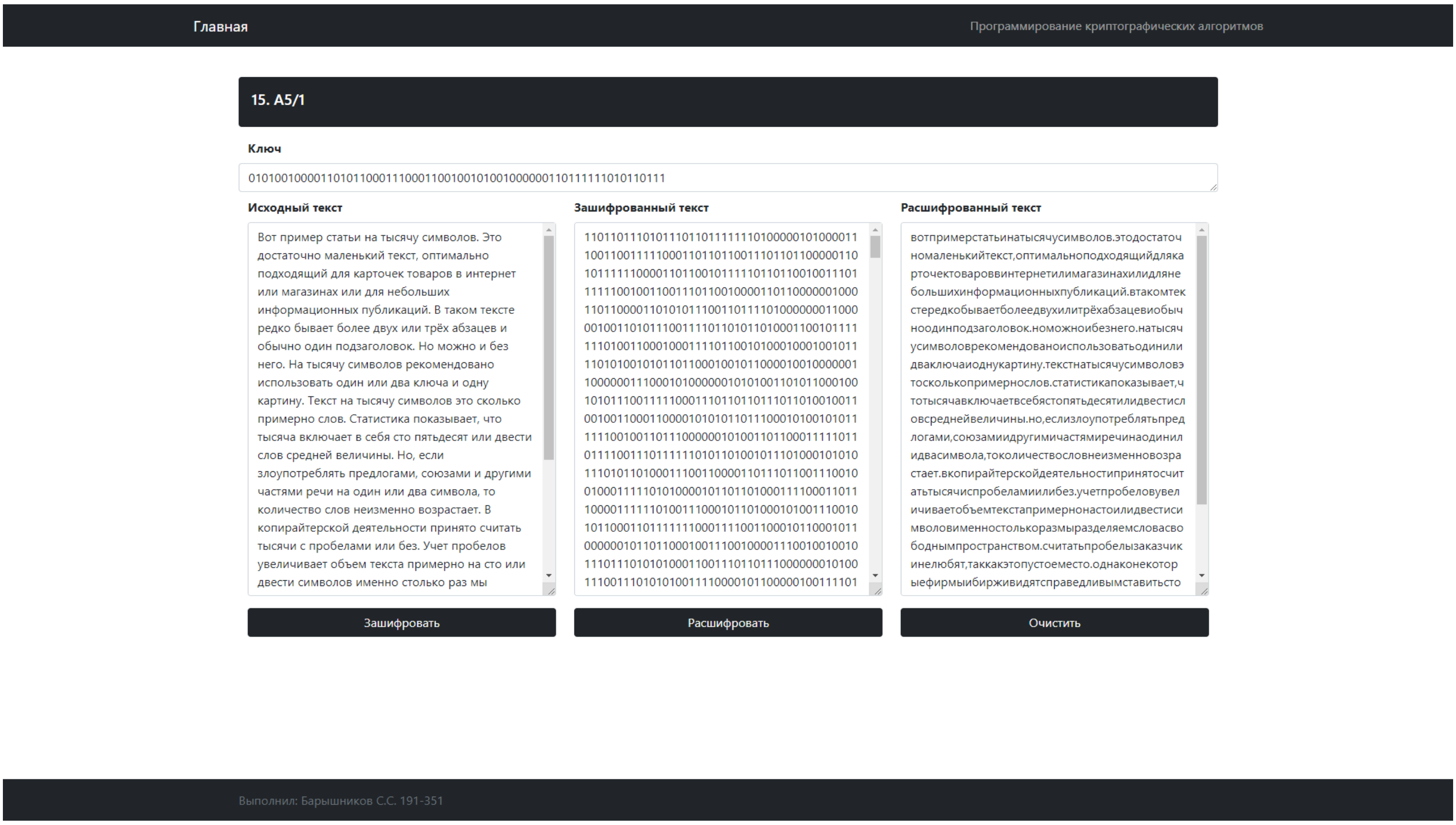
Тестирование:

```
/bin/python3
/root/mospolytech-education-crypt-dev-2021-1/lab06_15_a51.py

A5/1:
Ключ: 0101001000011010110001110001100100101001000000110111111010110111
КОРОТКИЙ ТЕКСТ:
Зашифрованный текст:
1101101110101110101000011101011100100000111001010111110010011101100100101
1011000001011011100001111101111010001101101100110111000001000100110001011
1100111111001100011101101101111101011010111000111111010011100110000010011
1010100101111011001000110011010100101110100101001000100010000101011001011
0010111110110010101111100010001010010100111111011000011000000101000010011
1010011000100000101010111001100001001101101110000110111001000010000100011
0000011010111011101100111101011111000

Расшифрованный текст:
времяприливыиотливынеждутчеловека.

ДЛИННЫЙ ТЕКСТ:
Зашифрованный текст:
1101101110101110110111111101000001010000111001100111110001101101100111011
1011000001101011111100001101100101111110110110010011101111111001001100111
1100100001101100000010001101100001101010111001101111010000000110000010011
1010111001111011010110100011001011111110100110001000111011001010001000100
0010111101010010101101100010010110000100100000011000000111000101000000010
1010011010110001001010111001111100011101101101110110100100110010011000011
0000101010110111000101001010111111001001101110000000101001101100011111011
0111100111011111101011010010111010001010101110101101000111001100001101110
0110011100100100011111010100001011011010001111000110111000011111101001110
0001011010001010011100101011000110111111100011110011000101100010110000000
1011011000100111001000011100100100101110111010101000110011101101110000000
```



```
111000111010110011110100001010001010101100010010001111011110010001011001
001011001101100001100000001000111000101100110100110011000011110110100010 1
001100010010111111100010010100000110110100111100011000111001011101010010
101110001010110011111111101000110111000011101010010110001101110011110111
000001001001001101010011111101001000111100010000010110111110111001000101
000001100011110100101010100111001001000101111000100101100011001011101111
001110100101010000000100000001000101011010110111010010110000000111111010111
000100101110001110110000101011001000010101100110000001101111001100111100
000100110011111110011011100100011011110011011010110010011101010001101100
111000101111010100000111001000100101101111000110001001101101000100110001
000001101001110111010111011010111011101001001100110000110010011111101010
101011011010

Расшифрованный текст:
вотпримерстатьинатысячусимволов.этодостаточномаленькийтекст,оптимальнопо
дходящийдлякарточектовароввинтернетилимагазинахилидлянебольшихинформацио
нныхпубликаций.втакомтекстередкобываетболеедвухилитрёхабзацевиобычнодинп
одзаголовок.номожноибезнего.натысячусимволоврекомендованоиспользоватьоди
нилидваключаиоднукартину.текстнатысячусимволовэтосколькопримернослов.ста
тистикапоказывает,чтотысячавключаетвсебястопятьдесятилидвестисловсредней
величины.но,еслизлоупотреблятьпредлогами,союзамиидругимичастямиречинаоди
нилидвасимвола,токоличествословнеизменновозрастает.вкопирайтерскойдеятел
ьностипринятосчитатьтысячиспробеламиилибез.учетпробеловувеличиваетобъемт
екстапримернонастоилидвестисимволовименностолькоразмыразделяемсловасвобо
днымпространством.считатьпробелызаказчикинелюбят,таккакэтопустоеместо.од
наконекоторыефирмыибирживидятсправедливымставитьстоимостьзатысячусимволо
вспробелами,считаяпоследниеважнымэлементомкачественноговосприятия.соглас
итесь,читатьслитныйтекстбезединогопропуска,никтонебудет.нобольшинствунуж
наценазатысячузнаковбезпробелов.
```

Интерфейс:



16.A5 /2

A5 — это поточный алгоритм шифрования, используемый для обеспечения конфиденциальности передаваемых данных между телефоном и базовой станцией в европейской системе мобильной цифровой связи GSM (Groupe Spécial Mobile).

Шифр основан на побитовом сложении по модулю два (булева операция «исключающее или») генерируемой псевдослучайной последовательности и шифруемой информации. В A5 псевдослучайная последовательность реализуется на основе трёх линейных регистров сдвига с обратной связью. Регистры имеют длины 19, 22 и 23 бита соответственно. Сдвигами управляет специальная схема, организующая на каждом шаге смещение как минимум двух регистров, что приводит к их неравномерному движению. Последовательность формируется путём операции «исключающее или» над выходными битами регистров.

Блок-схема:

Код программы:

```
# -*- coding:utf-8 -*-
# импорт компонентов, необходимых для работы программы
from base import alphabet, input_for_cipher_short,
input_for_cipher_long, output_from_decrypted
import sys
import copy
import re
```

```
reg_x_length = 19
reg_y_length = 22
reg_z_length = 23
reg_e_length = 17

key_one = ""
reg_x = []
reg_y = []
reg_z = []
reg_e = []

# функция загрузки регистров
def loading_registers(key):
    i = 0
    while(i < reg_x_length):
        reg_x.insert(i, int(key[i]))
        i = i + 1
    j = 0
    p = reg_x_length
    while(j < reg_y_length):
        reg_y.insert(j, int(key[p]))
        p = p + 1
        j = j + 1
    k = reg_y_length + reg_x_length
    r = 0
    while(r < reg_z_length):
        reg_z.insert(r, int(key[k]))
        k = k + 1
        r = r + 1
    i = 0
    while(i < reg_e_length):
        reg_e.insert(i, int(key[i]))
        i = i + 1

# функция установки ключа
def set_key(key):
    if(len(key) == 64 and re.match("^([01])+", key)):
        key_one = key
        loading_registers(key)
        return True
    return False

def get_key():
    return key_one

# функция перевода текста в бинарный код
def to_binary(plain):
    s = ""
    i = 0
    for i in plain:
        binary = str(' '.join(format(ord(x), 'b') for x in i))
        j = len(binary)
        while(j < 12):
```

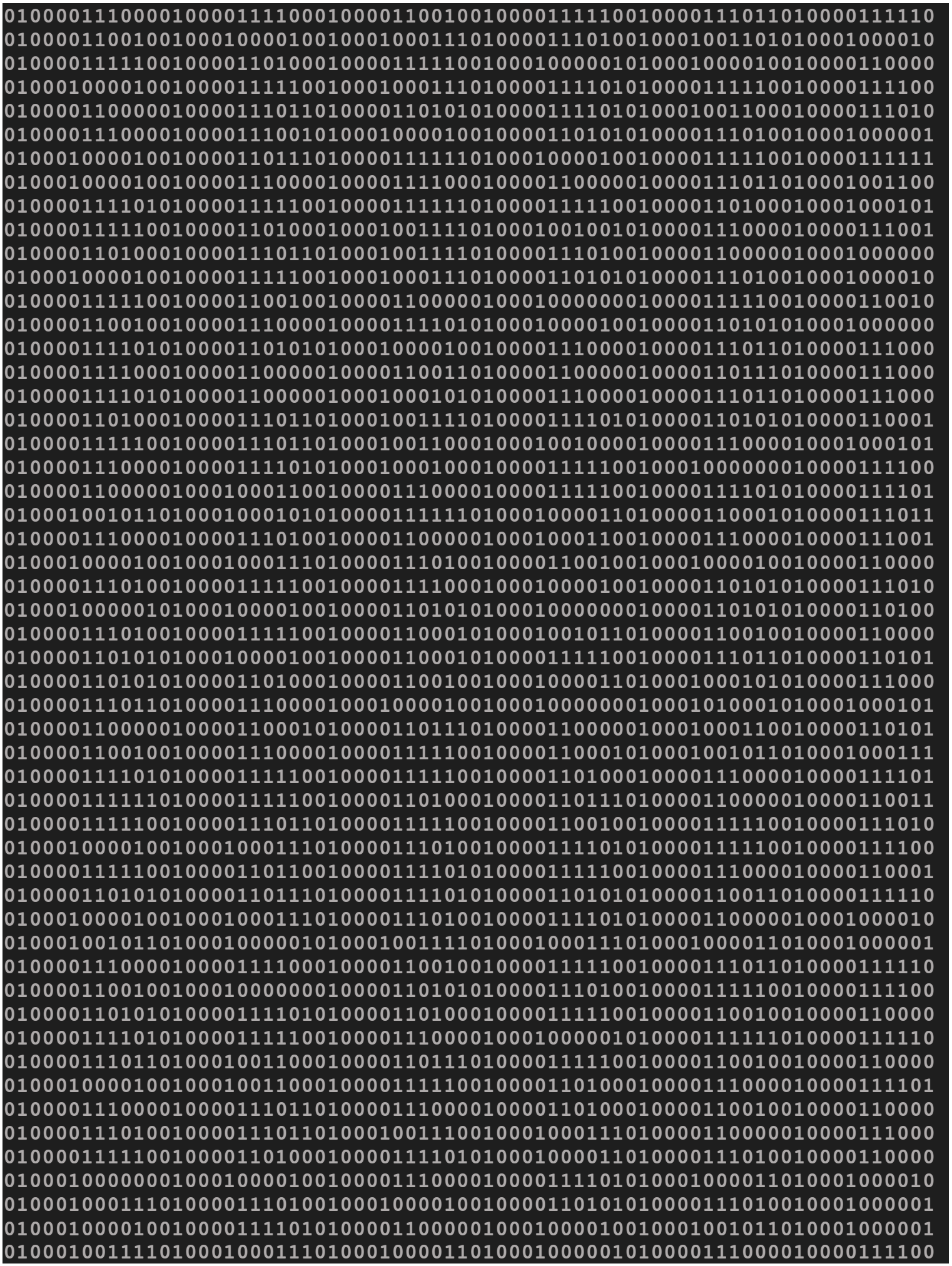
```
            binary = "0" + binary
            s = s + binary
            j = j + 1
    binary_values = []
    k = 0
    while(k < len(s)):
        binary_values.insert(k, int(s[k]))
        k = k + 1
    return binary_values

def get_majority(x, y, z):
    if(x + y + z > 1):
        return 1
    else:
        return 0

# функция получения потока
def get_keystream(length):
    reg_x_temp = copy.deepcopy(reg_x)
    reg_y_temp = copy.deepcopy(reg_y)
    reg_z_temp = copy.deepcopy(reg_z)
    reg_e_temp = copy.deepcopy(reg_e)
    keystream = []
    i = 0
    while i < length:
        majority = get_majority(reg_e_temp[3], reg_e_temp[7],
reg_e_temp[10])
        if get_majority(reg_x_temp[12], reg_x_temp[14], reg_x_temp[15])
== majority:
            new = reg_x_temp[13] ^ reg_x_temp[16] ^ reg_x_temp[17] ^
reg_x_temp[18]
            reg_x_temp_two = copy.deepcopy(reg_x_temp)
            j = 1
            while(j < len(reg_x_temp)):
                reg_x_temp[j] = reg_x_temp_two[j-1]
                j = j + 1
            reg_x_temp[0] = new

        if get_majority(reg_y_temp[9], reg_y_temp[13], reg_y_temp[16]) ==
majority:
            new_one = reg_y_temp[20] ^ reg_y_temp[21]
            reg_y_temp_two = copy.deepcopy(reg_y_temp)
            k = 1
            while(k < len(reg_y_temp)):
                reg_y_temp[k] = reg_y_temp_two[k-1]
                k = k + 1
            reg_y_temp[0] = new_one

        if get_majority(reg_z_temp[13], reg_z_temp[16], reg_z_temp[18])
== majority:
            new_two = reg_z_temp[7] ^ reg_z_temp[20] ^ reg_z_temp[21] ^
reg_z_temp[22]
            reg_z_temp_two = copy.deepcopy(reg_z_temp)
```

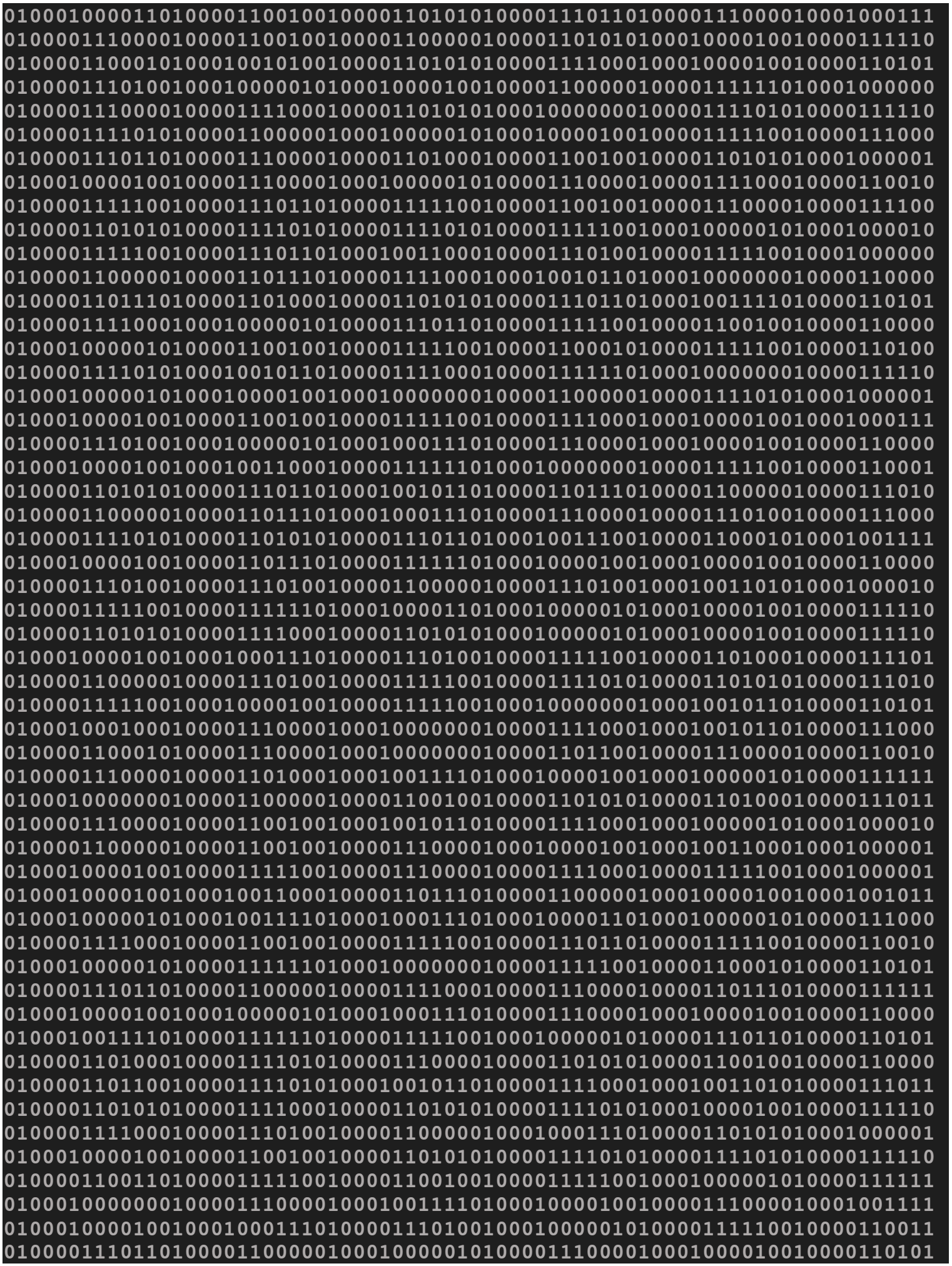
```
            m = 1
            while(m < len(reg_z_temp)):
                reg_z_temp[m] = reg_z_temp_two[m-1]
                m = m + 1
            reg_z_temp[0] = new_two

        keystream.insert(i, reg_x_temp[18] ^ reg_y_temp[21] ^
reg_z_temp[22])
        i = i + 1
    return keystream

# функция перевода бинарного кода в текст
def convert_binary_to_str(binary):
    s = ""
    length = len(binary) - 12
    i = 0
    while(i <= length):
        s = s + chr(int(binary[i:i+12], 2))
        i = i + 12
    return str(s)

# функция шифрования
def encrypt(plain):
    s = ""
    binary = to_binary(plain)
    keystream = get_keystream(len(binary))
    i = 0
    while(i < len(binary)):
        s = s + str(binary[i] ^ keystream[i])
        i = i + 1
    return s

# функция расшифрования
def decrypt(cipher):
    s = ""
    binary = []
    keystream = get_keystream(len(cipher))
    i = 0
    while(i < len(cipher)):
        binary.insert(i, int(cipher[i]))
        s = s + str(binary[i] ^ keystream[i])
        i = i + 1
    return convert_binary_to_str(str(s))

# функция проверки введенного ключа
def user_input_key():
    tha_key = str(input('Введите 64-bit ключ: '))
    if (len(tha_key) == 64 and re.match("^([01])+", tha_key)):
        return tha_key
    else:
        while(len(tha_key) != 64 and not re.match("^([01])+", tha_key)):
            if (len(tha_key) == 64 and re.match("^([01])+", tha_key)):
                return tha_key
```

```
            tha_key = str(input('Введите 64-bit ключ: '))
    return tha_key

# установка ключа
key = '0101001000011010110001110001100100101001000000110111111010110111'
set_key(key)

#вывод результатов работы программы
print(f'''
A5/2:
Ключ: {key}
КОРОТКИЙ ТЕКСТ:
Зашифрованный текст:
{encrypt(input_for_cipher_short())}

Расшифрованный текст:
{output_from_decrypted(decrypt(encrypt(
    input_for_cipher_short())))}

ДЛИННЫЙ ТЕКСТ:
Зашифрованный текст:
{encrypt(input_for_cipher_long())}

Расшифрованный текст:
{output_from_decrypted(decrypt(encrypt(
    input_for_cipher_long())))}
''')
```

Тестирование:

```
/bin/python3
/root/mospolytech-education-crypt-dev-2021-1/lab06_16_a52.py

A5/2:
Ключ: 0101001000011010110001110001100100101001000000110111111010110111
КОРОТКИЙ ТЕКСТ:
Зашифрованный текст:
0100001100100100010000000100001101010100001111000100010011110100000110111
0100001111110100010000100100001111111010001000000100001110000100001110110
0100001110000100001100100100010010110100001110000100001111100100010000100
0100001110110100001110000100001100100100010010110100001111010100001101010
0100001101100100001101000100010000110100010000100100010001110100001101010
0100001110110100001111100100001100100100001101010100001110100100001100000
0100010000100100010001110100001110100
Расшифрованный текст:
время,приливыиотливынеждутчеловека.

ДЛИННЫЙ ТЕКСТ:
Зашифрованный текст:
0100001100100100001111100100010000100100001111111010001000000100001110000
0100001111000100001101010100010000000100010000010100010000100100001100000
0100010000100100010011000100001110000100001111010100001100000100010000100
0100010010110100010000010100010011110100010001110100010000110100010000001
```




01000100001101000011001001000011010101000011101101000011100001000100011
01000011100001000011001001000011000001000011010101000100001001000011110
01000011000101000100101001000011010101000011110001000100001001000011010
01000011101001000100000101000100001001000011000001000011111101000100000
01000011100001000011110001000011010101000100000001000011110101000011110
01000011110101000011000001000100000101000100000100100001111100100001110
01000011101101000011100001000011010001000011001001000011010101000100001
01000100001001000011100001000100000101000011100001000011110001000011010
01000011111001000011101101000111110010000110010010000111000010000111100
01000011010101000011110101000011110100000111110010000111100010000110100
01000011111001000011101101000100110001000011101001000011111001000100000
01000011000001000011011101000011110001000100101101000100000001000011000
01000011011101000011010001000011010101000011101101000100111101000011010
01000011110001000100000101000100001111111010001000000100001111010000110
01000100000101000100001001000100000001000011000001000011110101000100001
01000100001001000011001001000011111001000011100010100010000111100010100
01000011101001000100000101000100011101000011100001000100001001000011000
01000100001001000100110001000011111101000100000001000011111001000011000
01000011010101000011101101000100101101000011011101000011000001000011101
01000011000001000011101001000011111001000011110101000011010101000011010
01000011111001000100001001000011111001000100000001000100101101000011010
01000011010101000011110001000011010101000100000101000100001001000011110
01000100001001000011101001000011111001000011110101000011010101000011010
01000011000001000011101001000011111001000011110101000011010101000011101
01000100001001000011101001000011111001000100000001000100101101000011100
01000100001001000100110001000011011101000011000001000100001001000100101
01000100001010000100111101000100011101000100000110100010000010100001110
01000011110001000011001001000011111001000011101101000011111001000011010
01000100000101000011111101000100000001000011111001000011000101000011010
01000011101101000011100001000011010101000011110101000100001001000011111
01000011010101000011110001000011010101000011110101000100001001000011110
01000011110001000011101001000011000001000100011101000011010101000100001
01000100001001000011001001000011010101000011110101000011110101000011110
01000011001101000011111001000011001001000011111001000100000101000011111
01000100000001000011100001000100111101000100000100100001110000100010011
01000100001001000100011101000011101001000100000101000011111001000011011
01000011101101000011000001000100000101000011100001000100001001000011010

```
0100010000010100010011000100001101110100001111110100010000100100010001111
0100001110000100010000100100001100000100010000100100010011000100010000001
0100001110110100001110000100010000100100001111010100010010110100000111001
0100010000100100001101010100001110100100010000010100010000100100000110001
0100001101010100001101110100001101010100001101000100001110000100001111101
0100001111100100001100110100001111100100001111111010001000000100001111110
0100001111110100010000110100010000010100001110100100001100000100001101111
0100001111110100010000100100001111010100001110000100001110100100010000010
0100001111100100001111010100001101010100001100010100010000110100001101000
0100001101010100010000100100010000100100010001110100001110100100001111101
0100001111100100001100010100001111100100001110110100010011000100010010000
0100001110000100001111010100010000010100010000100100001100100100010000011
0100001111010100010000110100001101100100001111010100001100000100010001110
0100001101010100001111010100001100000100001101110100001100000100010000010
0100100101101000100000101000100111101000100011101000100001101000011011l
0100001111010100001100000100001110100100001111100100001100100100001100001
0100001101010100001101110100001111110100010000000100001111100100001100001
0100001101010100001110110100001111100100001100100100010000100100010001111
010000111010
```

Расшифрованный текст:

вотпримерстатьинатысячусимволов.этодостаточномаленькийтекст,оптимальнопо
дходящийдлякарточектовароввинтернетилимагазинахилидлянебольшихинформацио
нныхпубликаций.втакомтекстередкобываетболеедвухилитрёхабзацевиобычноодинп
одзаголовок.номожноибезнего.натысячусимволоврекомендованоиспользоватьоди
нилидваключаиоднукартину.текстнатысячусимволовэтосколькопримернослов.ста
тистикапоказывает,чтотысячавключаетвсебястопятьдесятилидвестисловсредней
величины.но,еслизлоупотреблятьпредлогами,союзамиидругимичастямиречинаоди
нилидвасимвола,токоличествословнеизменновозрастает.вкопирайтерскойдеятел
ьностипринятосчитатьтысячиспробеламиилибез.учетпробеловувеличиваетобъемт
екстапримернонастоилидвестисимволовиименностолькоразмыразделяемсловасвобо
днымпространством.считатьпробелызаказчикинелюбят,таккакэтопустоеместо.од
наконекоторыефирмыибиржividятсправедливымставитьстоимостьзатысячусимволо
вспробелами,считаяпоследниеважнымэлементомкачественноговосприятия.соглас
итесь,читатьслитныйтекстбезединогопропуска,никтонебудет.нобольшинствунуж
наценазатысячузнаковбезпробелов.

Интерфейс:

15. A5/2

Ключ

0101001000011010110001110001100100101001000000110111111010110111

Исходный текст

Вот пример статьи на тысячу символов. Это достаточно маленький текст, оптимально подходящий для карточек товаров в интернет или магазинах или для небольших информационных публикаций. В таком тексте редко бывает более двух или трёх абзацев и обычно один подзаголовок. Но можно и без него. На тысячу символов рекомендовано использовать один или два ключа и одну картину. Текст на тысячу символов это сколько примерно слов. Статистика показывает, что тысяча включает в себя сто пятьдесят или двести слов средней величины. Но, если злоупотреблять предлогами, союзами и другими частями речи на один или два символа, то количество слов неизменно возрастает. В копирайтерской деятельности принято считать тысячи с пробелами или без. Учет пробелов увеличивает объем текста примерно на сто или двести символов именно столько раз мы

Зашифрованный текст

0100001100100100001111100100010000100100000
1111110100010000000100001110000100001111000
0100001101010100010000000100010000010100010
0000100100001100000100010000100100010011000
0100001110000100001111010100001100000100010
0000100100010010110100010000010100010011110
0100010001110100010000110100010000010100000
1110000100001111000100001100100100001111100
0100001110110100001111100100001100100100010
0000100100010001110100001110100100010011010
0100010000100100001111100100010000100100000
1111100100010000010100010000100100001100000
0100010000100100001111100100010001110100000
1111010100001111100100001111000100001100000
0100001110110100001101010100001111010100010
0011000100001110100100001110000100001110010
0100010000100100001101010100001110100100010
0000010100010000100100001101110100001111110
0100010000100100001111100100001111110100010
0000100100001110000100001111000100001100000

Расшифрованный текст

вотпримерстатьинатысячусимволов.этодостаточ
номаленькийтекст,оптимальноподходящийдляка
рточектовароввинтернетилимагазинахилидляне
большихинформационныхпубликаций.втакомтек
стередкобываетболеедвухилитрёхабзацевиобыч
ноодинподзаголовок.номожноибезнего.натысяч
усимволоврекомендованоиспользоватьодинили
двакючаиоднукартину.текстнатысячусимволовэ
тосколькопримернослов.статистикапоказывает,ч
тотысячавключаетвсебястопятьдесятилидвестисл
овсреднейвеличины.но,еслизлоупотреблятьпред
логами,союзамиидругимичастямиречинаодинил
идвасимвола,токоличествословнеизменновозра
стает.вкопирайтерскойдеятельностипринятосчит
атьтысячиспробеламиилибез.учетпробеловувел
ичиваетобъемтекстапримернонастоилидвестиси
мволовименностолькоразмыразделяемсловасво
боднымпространством.считатьпробелызаказчик
инелюбят,таккакэтопустоеместо.однаконекотор
ыефирмыибиржividятсправедливымставитьсто

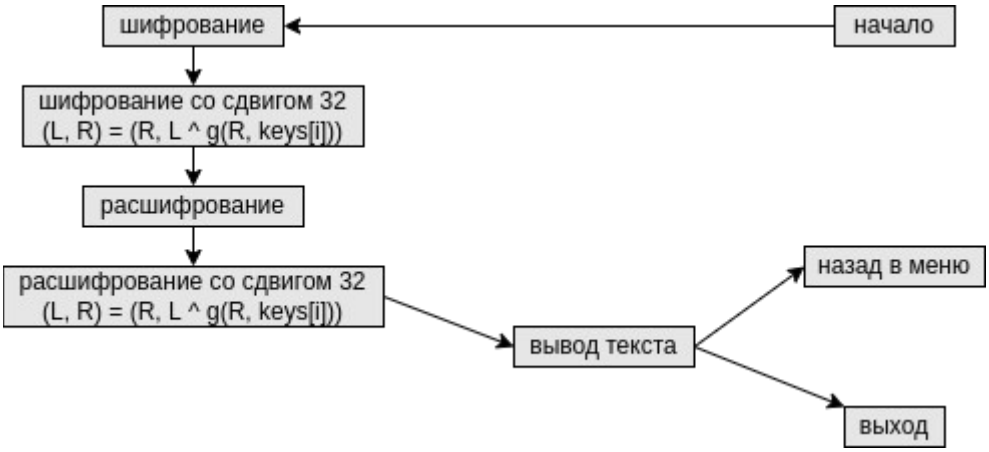
Зашифровать | Расшифровать | Очистить

Блок G: КОМБИНАЦИОННЫЕ ШИФРЫ

17.МАГМА

Магма представляет собой симметричный блочный алгоритм шифрования с размером блока входных данных 64 бита, секретным ключом 256 бит и 32 раундами шифрования.

Блок-схема:



Код программы:

```
# импорт компонентов, необходимых для работы программы
from base import alphabet, input_for_cipher_short,
input_for_cipher_long, output_from_decrypted

pi0 = [12, 4, 6, 2, 10, 5, 11, 9, 14, 8, 13, 7, 0, 3, 15, 1]
pi1 = [6, 8, 2, 3, 9, 10, 5, 12, 1, 14, 4, 7, 11, 13, 0, 15]
pi2 = [11, 3, 5, 8, 2, 15, 10, 13, 14, 1, 7, 4, 12, 9, 6, 0]
pi3 = [12, 8, 2, 1, 13, 4, 15, 6, 7, 0, 10, 5, 3, 14, 9, 11]
pi4 = [7, 15, 5, 10, 8, 1, 6, 13, 0, 9, 3, 14, 11, 4, 2, 12]
pi5 = [5, 13, 15, 6, 9, 2, 12, 10, 11, 7, 8, 1, 4, 3, 14, 0]
pi6 = [8, 14, 2, 5, 6, 9, 1, 12, 15, 4, 11, 0, 13, 10, 3, 7]
pi7 = [1, 7, 14, 13, 0, 5, 8, 3, 4, 15, 10, 6, 9, 12, 11, 2]

pi = [pi0, pi1, pi2, pi3, pi4, pi5, pi6, pi7]

MASK32 = 2 ** 32 - 1

def t(x):
    y = 0
    for i in reversed(range(8)):
        j = (x >> 4 * i) & 0xf
        y <<= 4
        y ^= pi[i][j]
    return y

# функция сдвига на 11
def rot11(x):
    return ((x << 11) ^ (x >> (32 - 11))) & MASK32
```

```
def g(x, k):
    return rot11(t((x + k) % 2 ** 32))

def split(x):
    L = x >> 32
    R = x & MASK32
    return (L, R)

def join(L, R):
    return (L << 32) ^ R

def magma_key_schedule(k):
    keys = []
    for i in reversed(range(8)):
        keys.append((k >> (32 * i)) & MASK32)
    for i in range(8):
        keys.append(keys[i])
    for i in range(8):
        keys.append(keys[i])
    for i in reversed(range(8)):
        keys.append(keys[i])
    return keys

# функция шифрования
def magma_encrypt(x, k):
    keys = magma_key_schedule(k)
    (L, R) = split(x)
    for i in range(31):
        (L, R) = (R, L ^ g(R, keys[i]))
    return join(L ^ g(R, keys[-1]), R)

# функция расшифрования
def magma_decrypt(x, k):
    keys = magma_key_schedule(k)
    keys.reverse()
    (L, R) = split(x)
    for i in range(31):
        (L, R) = (R, L ^ g(R, keys[i]))
    return join(L ^ g(R, keys[-1]), R)

# установка ключа
key =
int('ffeeddccbbaa99887766554433221100f0f1f2f3f4f5f6f7f8f9fafbfcfdfeff',
16)

i = 0
text_short = input_for_cipher_short()
encr_short = []
while (i < len(text_short)):
    text = text_short[i:i+4].encode().hex()
    text = int(text, 16)
    text = text % 2**64
    pt = text
```

```
    ct = magma_encrypt(pt, key)
    encr_short.append(ct)
    i += 4
decr_short = []
for i in encr_short:
    dt = magma_decrypt(i, key)
    decr_short.append(bytes.fromhex(hex(dt)[2::]).decode('utf-8'))

i = 0
text_long = input_for_cipher_long()
encr_long = []
while (i < len(text_long)):
    text = text_long[i:i+4].encode().hex()
    text = int(text, 16)
    text = text % 2**64
    pt = text
    ct = magma_encrypt(pt, key)
    encr_long.append(ct)
    i += 4
decr_long = []
for i in encr_long:
    dt = magma_decrypt(i, key)
    decr_long.append(bytes.fromhex(hex(dt)[2::]).decode('utf-8'))

#вывод результатов работы программы
print(f'''
МАГМА:
КЛЮЧ:
{key}

КОРОТКИЙ ТЕКСТ:
Зашифрованный текст:
{encr_short}

Расшифрованный текст:
{output_from_decrypted(''.join(decr_short))}

ДЛИННЫЙ ТЕКСТ:
Зашифрованный текст:
{encr_long}

Расшифрованный текст:
{output_from_decrypted(''.join(decr_long))}
''')
```

Тестирование:

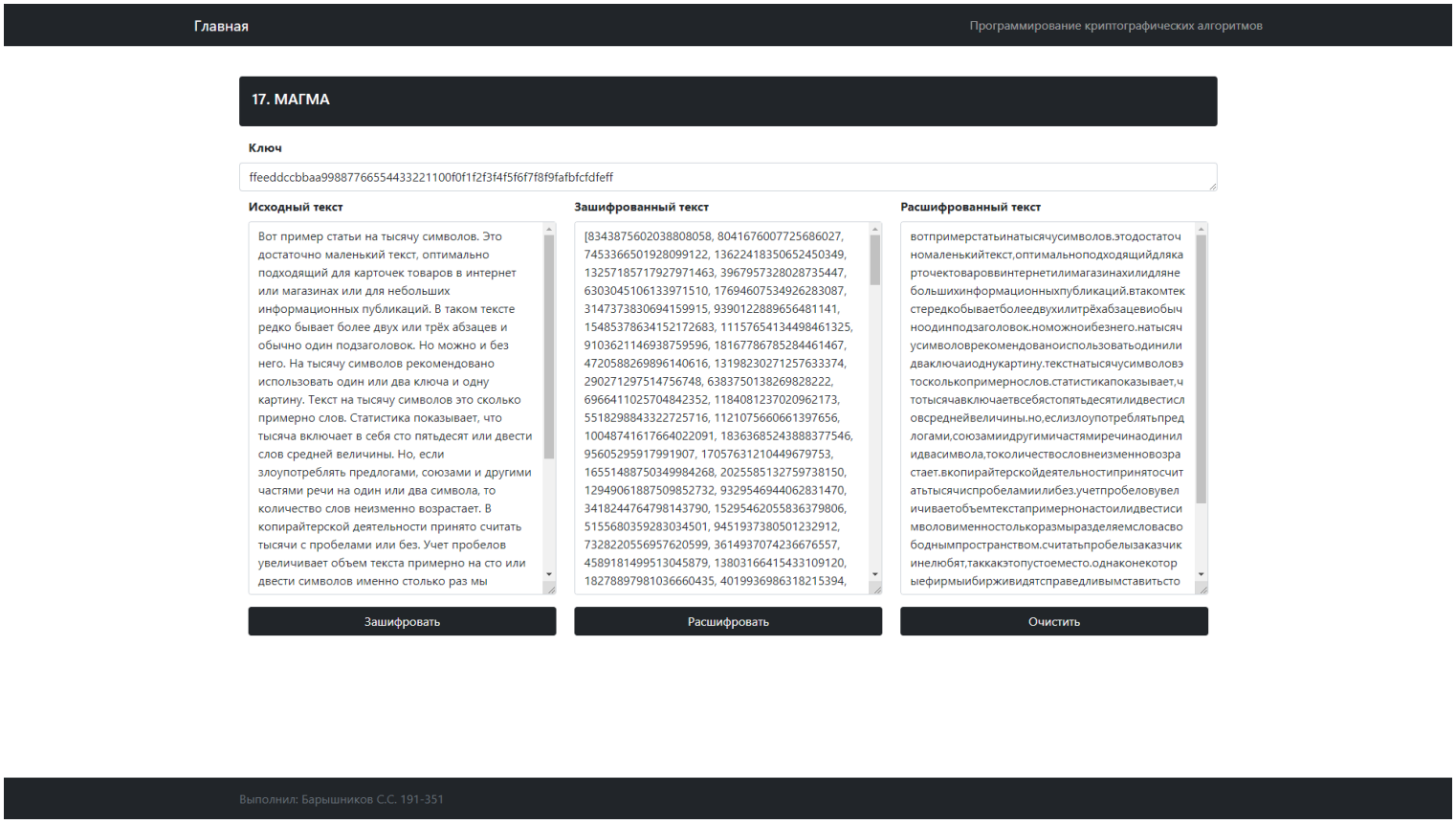
```
/bin/python3
/root/mospolytech-education-crypt-dev-2021-1/lab07_17_magma.py

МАГМА:
КЛЮЧ:
```

```
1157618167956855245228066527250255057862004105058474443086885538920014061
23775

КОРОТКИЙ ТЕКСТ:
Зашифрованный текст:
[18432907413224455314, 10996816857283808610, 1603220777717569738,
67983393742724255273, 6252753798785705582, 12897841916972840738,
12693135464871535956, 6338906346771095526, 2952080121925535959,
10585345143535530769]

Расшифрованный текст:
время,приливыиотливынеждутчеловека.

ДЛИННЫЙ ТЕКСТ:
Зашифрованный текст:
[83438756020388080058, 8041676007725686027, 7453366501928099122,
13622418350652450349, 13257185717927971463, 39679573280287354447,
63030451061339715010, 17694607534926283087, 31473738306941599115,
93901228896564811141, 15485378634152172683, 11157654134498461325,
91036211469387595960, 18167786785284461467, 4720588269896140616,
13198230271257633374, 2902712975147567480, 63837501382698282222,
69664110257048423520, 11840812370209621730, 55182988433227257160,
11210756606613976560, 10048741617664022091, 18363685243888377546,
956052959179919070, 170576312104496797530, 165514887503499842680,
20255851327597381500, 129490618875098527320, 93295469440628314700,
34182447647981437900, 152954620558363798060, 51556803592830345010,
94519373805012329120, 73282205569576205990, 36149370742366765570,
45891814995130458790, 13803166415433109120, 18278897981036660435,
40199369863182153940, 12394731635219602309, 54162488580339238160,
46323282895002593320, 159897598345341780490, 133609392736749150280,
85286085552919060000, 102934537315336503330, 132389754150126789980,
12616809652721693680, 104838295245852491950, 157035603361047735810,
29286127887899747860, 123707193479695661680, 154980188456511244010,
79752020056154359220, 10016402103949689310, 94823908768763094600,
10306886503249707528, 34323711200831045760, 132847235221599841340,
47329579363084845480, 181451500746987974210, 76659126580164409550,
17186319045640305491, 13257185717927971463, 39679573280287354470,
63030451061339715010, 11637202970910093420, 23466997127566948090,
10356924971308547229, 139907356427995025460, 166568373037217771900,
36209708847117289770, 94434872926303641510, 76910490946820269090,
916112073177174940, 157974669419969022290, 167775099285854896390,
63031214702137987110, 116748848993958131900, 180465744290070171280,
13077766683806041775, 65692148579544394280, 13257185717927971463,
39679573280287354470, 63030451061339715010, 181310682781868585130,
160024180939452990270, 29081148687787074520, 117943926020891014670,
11996918218642771910, 30769383504769500060, 134508174714865943790,
10728741519359495825, 92691896824800915970, 141117782682980964190,
20578963828445017040, 102934537315336503330, 172042124216868487390,
53662062268688007020, 82484669974485367220, 100888105659572758500,
22543550902437422570, 83718968843992981330, 109855183325432858510,
17941711041904849701, 77402002079432793540, 107504195515386382090,
26292977765199590890, 22257420270936085020, 28724497422719375100,
17712479779128577940, 92222969970569946150, 973788500067181560,
```

3393954245381334839, 10641483480207737426, 16269104980183698483,
16250235473790191753, 2700688729269881630, 84116078843487540 50,
7636448048795956798, 692746442652315515, 113430705279078739 16,
629521920599350430, 14623128418242633989, 5705247633908105959,
17910568456840661856, 16201319981229519019, 137692988824631797 21,
7421154077805485456, 9312183877157555560, 769104909468202690 9,
916112073177174944, 15135398776342705892, 152892603731320816 71,
71263790601415682 92, 17250369459672726 73, 176887558563542971 23,
138113508643393643, 14127475102588350875, 4151016356341159175,
98414674873937416 36, 31879593274959246 97, 157280813244224474 59,
16055605958376054530, 3572266014017346359, 32238270005876804 74,
82537335001663112 51, 91405085422878099 51, 17900326362396929 33,
14268543881935307321, 73424823037920083 83, 176775709639087814 00,
13355264069525559 38, 88597409000583480, 172570301845273904 06,
12715322944275189718, 43113056607886706 62, 70084620566287577 49,
5705247633908105959, 780471545777017295, 718133742958491815 4,
17602858510462705198, 147380618154847797 55, 540299310016161264,
10182683375894303875, 80263432045513379 82, 176603789405815346 82,
13825198160625090606, 47205882698961406 16, 57305274094012220 56,
17941293450735671408, 43423663519841470 86, 531327644815816537 9,
17450241683162283 46, 25281365199649886 65, 35477558118573890 63,
10306886503249707528, 45264948651592282 42, 73424823037920083 83,
29081148687780745 20, 74716800385967772 70, 164127993490393103 06,
17456146687817481297, 28943214759462716 48, 69231581617379714 79,
10027397076122517884, 11785376343832896276, 45488898444180685 35,
90681861366049598 43, 17006372751687582 512, 95822368420437095 84,
27563461970164099 68, 12319862187853716832, 147380618154847797 55,
35979860151276922 57, 60062309095022379 40, 80490890390167177 89,
61570471940346132 33, 69583740526642441 97, 164314788852936812 93,
89350112170546411 03, 44938110084128077 67, 29508217835671468 96,
18551799614657924 76, 67145244092806490 05, 135708180251221981 73,
14348383310813638904, 44530672786483204 96, 34382449849322553 10,
91080462400421436 94, 15127037950488644 1, 101839466942844108 05,
58025664584620513 0, 53564527649855059 55, 19495840053058447 60,
50545096165965068 97, 80656194706677404 8, 142326146259823777 49,
44864024590894814 20, 61420673747205419 17, 11774712973292423 210,
10305629430321463988, 35477558118573890 63, 10306886503249707528,
15489555906471766183, 15441006425873291865, 26598050834085520 87,
12582666581260422456, 54188264939658812 8, 48866090864426696 21,
55611968567671419 65, 77313258883951477 45, 10966269065365886 42,
76394760436279367 4, 127987947804960527 12, 82992425697894351 68,
67660617073868317 62, 89715285509582270 67, 112541813019370160 96,
41067934786911701 80, 58123053744651110 22, 141224312083969726 45,
87568808396852895 05, 18401452094994769 46, 172042124216868487 39,
12702832949266190125, 12312436585117905032, 27311611900111582 63,
47205882698961406 16, 13119149454917978330, 98221088514466533 48,
40856774283330940 07, 12601379616160262740, 102650873334626414 78,
87709561803165343 33, 27462124300767568 19, 107808030505526381 65,
19061242771921034 47, 83187902879887032 11, 94519373805012329 12,
71540105892017162 26, 65932387259159398 56, 163819602570935064 21,
16805496448556233763, 13257185717927971463, 35342388718972839 87,
18436104050552264826, 32371662915203870 28, 147380618154847797 55,
5402993100161612 64, 10585345143535530769]

```
Расшифрованный текст:
вотпримерстатьинатысячусимволов.этодостаточномаленькийтекст,оптимальнопо
дходящийдлякарточектоваровваинтернетилимагазинахилидлянебольшихинформацио
нныхпубликаций.втакомтекстередкобываетболеедвухилитрёхабзацевиобычноодинп
одзаголовок.номожноибезнего.натысячусимволоврекомендованоиспользоватьоди
нилидваключаиоднукартину.текстнатысячусимволовэтосколькопримернослов.ста
тистикапоказывает,чтотысячавключаетвсебястопятьдесятилидвестисловсредней
величины.но,еслизлоупотреблятьпредлогами,союзамиидругимичастямиречинаоди
нилидвасимвола,токоличествословнеизменновозрастает.вкопирайтерскойдеятел
ьностипринятосчитатьтысячиспробеламиилибез.учетпробеловувеличиваетобъемт
екстапримернонастоилидвестисимволовименностолькоразмыразделяемсловасвобо
днымпространством.считатьпробелызаказчикинелюбят,таккакэтопустоеместо.од
наконекоторыефирмыибиржividятсправедливымставитьстоимостьзатысячусимволо
вспробелами,считаяпоследниеважнымэлементомкачественноговосприятия.соглас
итесь,читатьслитныйтекстбезединогопропуска,никтонебудет.нобольшинствунуж
наценазатысячузнаковбезпробелов.
```

Интерфейс:

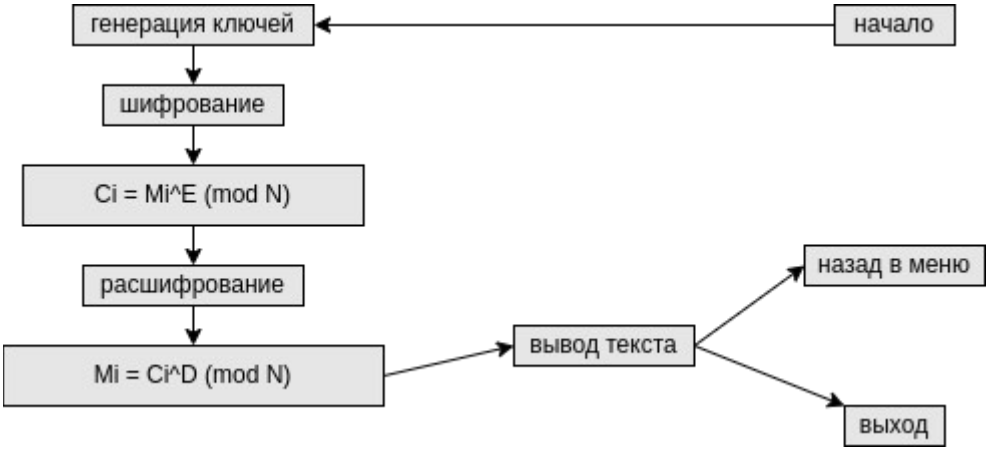


БЛОК H: АСИММЕТРИЧНЫЕ ШИФРЫ

21.RSA

Берутся два очень больших простых числа P и Q и находятся произведение простых чисел N=P×Q и функция Эйлера от этого произведения φ(N)=(P-1)×(Q-1). Выбирается случайное целое число E, взаимно простое с φ(N), и вычисляется D=(1 MOD φ(N))/E. E и N публикуются как открытый ключ, D сохраняется в тайне. Шифрование: Если M — сообщение, то шифртекст Ci получается последовательным шифрованием каждой шифрвеличины Mi возведением ее в степень E по модулю N: Ci = Mi EMOD N. Расшифрование: Получатель расшифровывает сообщение, возводя последовательно Ci в степень D по модулю N: Mi = Ci D MOD N.

Блок-схема:



Код программы:

```
# -*- coding:utf-8 -*-
# импорт компонентов, необходимых для работы программы
from base import alphabet, input_for_cipher_short,
input_for_cipher_long, output_from_decrypted
import random

# функция вычисления НОД
def gcd(a, b):
    while b != 0:
        a, b = b, a % b
    return a

# функция получения обратного числа
def multiplicative_inverse(e, r):
    for i in range(r):
        if((e*i) % r == 1):
            return i

# функция проверки числа на простоту
def is_prime(num):
    if num == 2:
        return True
```

```
    if num < 2 or num % 2 == 0:
        return False
    for n in range(3, int(num**0.5)+2, 2):
        if num % n == 0:
            return False
    return True

# функция генерации пары ключей
def generate_keypair(p, q):
    if not (is_prime(p) and is_prime(q)):
        raise ValueError('Оба числа должны быть простыми.')
    elif p == q:
        raise ValueError('p и q не могут быть равны друг другу')
    n = p * q

    phi = (p-1) * (q-1)

    e = random.randrange(1, phi)

    g = gcd(e, phi)
    while g != 1:
        e = random.randrange(1, phi)
        g = gcd(e, phi)
    d = multiplicative_inverse(e, phi)
    return ((e, n), (d, n))

# функция шифрования
def encrypt(pk, plaintext):
    key, n = pk
    cipher = [(ord(char) ** key) % n for char in plaintext]
    return cipher

# функция расшифрования
def decrypt(pk, ciphertext):
    key, n = pk
    plain = [chr((char ** key) % n) for char in ciphertext]
    return ''.join(plain)

# установка ключа
p = 107
q = 109
public, private = generate_keypair(p, q)

message_short = input_for_cipher_short()
encrypted_short = encrypt(private, message_short)
print_enc_short = ''.join([str(x) for x in encrypted_short])
decrypted_short = decrypt(public, encrypted_short)

message_long = input_for_cipher_long()
encrypted_long = encrypt(private, message_long)
print_enc_long = ''.join([str(x) for x in encrypted_long])
decrypted_long = decrypt(public, encrypted_long)
```

```
#вывод результатов работы программы
print(f'''
RSA:
Ключ:
p={p} q={q}
Публичный: {public}
Приватный: {private}
КОРОТКИЙ ТЕКСТ:
Зашифрованный текст:
{print_enc_short}

Расшифрованный текст:
{output_from_decrypted(decrypted_short)}

ДЛИННЫЙ ТЕКСТ:
Зашифрованный текст:
{print_enc_long}

Расшифрованный текст:
{output_from_decrypted(decrypted_long)}
''')
```

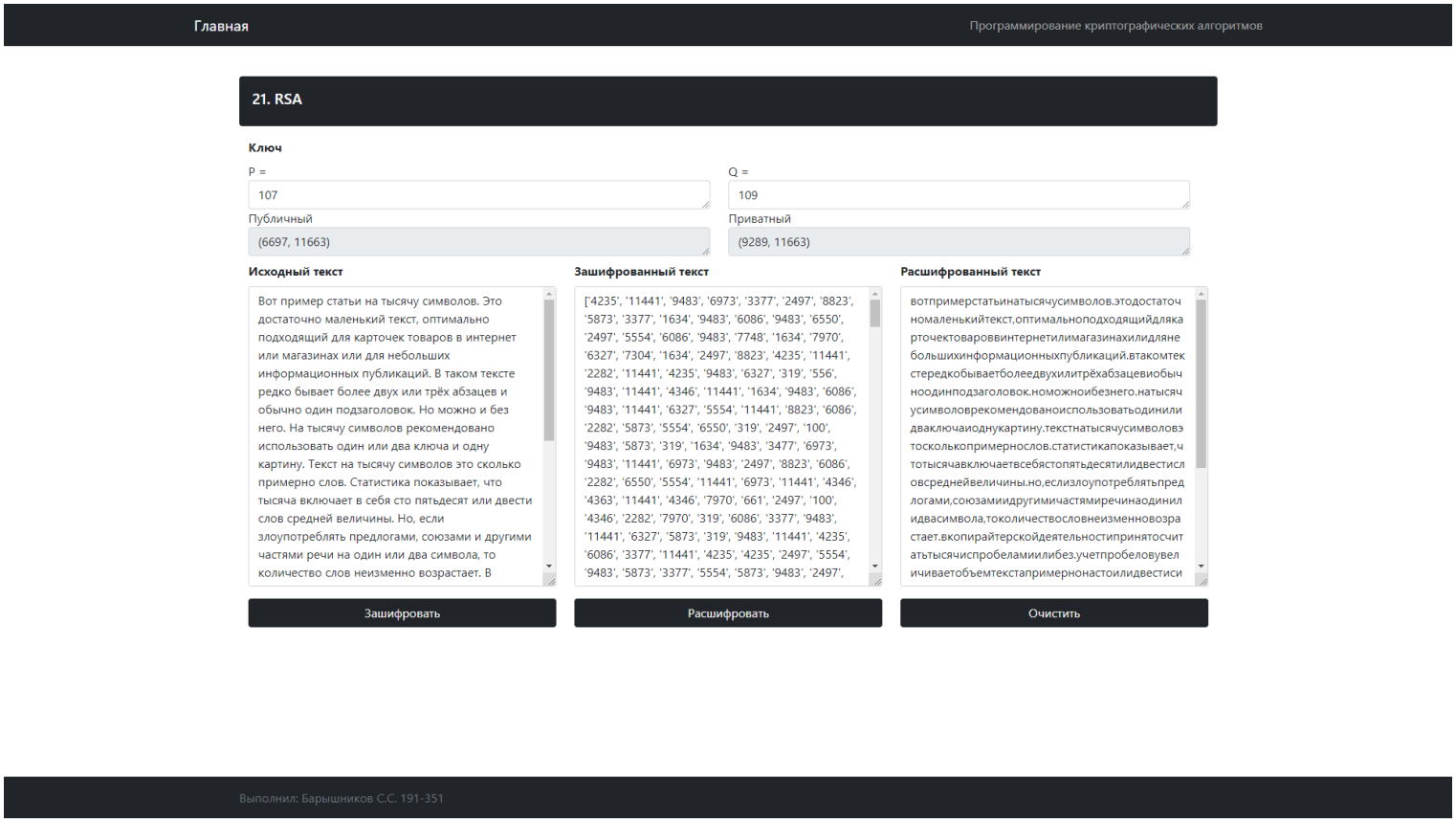
Тестирование:

```
/bin/python3
/root/mospolytech-education-crypt-dev-2021-1/lab08_21_rsa.py
└┘


RSA:
Ключ:
p=107 q=109
Публичный: (5441, 11663)
Приватный: (4025, 11663)
КОРОТКИЙ ТЕКСТ:
Зашифрованный текст:
75927171884793859901301256591002856597171188180418875921065418863381002818041887592106548477884731961075410029100286456884718046338759288472548100181002864562548

Расшифрованный текст:
время,приливыиотливынеждутчеловека.

ДЛИННЫЙ ТЕКСТ:
Зашифрованный текст:
75926338100285659717118893858847717121791002810018100284390188847710018100281065421799901645610029217918893857592633818046338759210028645625482656100286338107546338217910028100181002863386456847763389385100181804884784774390254818865681002888472548217910028301256591002863385659100281889385100181804439084776338565963381075432796338107549901669818865681075418049901254810018717110028633864568847254810028633875921001871716338759275921888477100288847717184778847100281881804188938510018653310018301218884771001832791881804188107541804990184778847310763381804439041311883279188847785256338717193851001894671886338847784771065432795659100293107180418825481001894671886568100286456254875921002810018254863389385100288847254 8
```

2179100288847717188471075425486338310710654759210018884710028310763381804884788471075475921002932791881804188100287171600232791001831073012100189467884775921886338310710654645684776338633810754188847756596338107543012100186533633818046338759263382548100286456254884776338938563383196847763381883107884730128477884765336338100286456254884771001810028106542179990164561002921791889385759263381804633875927171884725486338938588478471771075463387592100188477633818821795659633818044390301263387592100181002843906338107541888477188180418810754759210018254818041151964561001818863381075484771002925481001871711002818884771002910028645625481002888472548217910028810028106542179990164561002921791889385759263381804633875922654863385659717118893858847717184776338217918046338759210028645625482179100281001810028188217910028188254810018565963382548100183012106547592100188847100283012565910028645610028633810028106542179990164561001875922548180411519645610018884710028759221798847310799012179100286338565999011002843901075488472179990110028188180418810754759288472179100281882179180463387592217971718847107548477884765687592884718041886456188847710654100286456254884776338301256591002888472179180418830121804633810029565963381002871718847310718049901100284390565971718847107541804633865331001893851883012565910028217963381151930121001893851881881075471711002965331889385188645610018217910028990193851887171884764561888477100186338107541888477188180418810754759210018217918893857592633818041001830125659100281002863382548633818041886456884721791002875926338217918046338759284778847188301293858847847784776338759263383012717110018217910028100188847100281002864562548759225486338565918871711001865681002888477171217925486338656810754884799011002888471804439084776338217910028188565971711888477990110028633821796456188100281001810028439010028106542179990164561882179565971716338310788471804100189385188188180418831078847301210028645625481002964568847100285659717163383107884718046338759210029759288471804188645618875921001888471002863383107562788479385100288847254821791002810018565971711889385884771718477633884771001821791002863381881804188107547592884721791002818821791889385759263381804633875921889385884784778477633821791002863381804439025486338717110018301293851065471711001830121075488471804990188479385217918046338759210018217975926338310763381075484771065493855659717163382179100287171100188477217910028759263389385100286456254821796456188100281001810028439056597171633831078847180410654301210018254810018301264561882548188847788471804115193107990110028301256591002810028100182548254810018254826561002863385659100292179100286338884793858847217910028633810028645625486338107548477100182548633884778847254863381002863387171106548847852518871719385106541883107188717131961887592188107549901100282179565971711001875928847107541804188759210654938521791002810018759218810028439021791002863381889385633821791002843903012100181002810654217999016456100292179188938575926338180463387592217956597171633831078847180410018938518830125659100282179645618810028100189901565963382179180488471075484771888847759210018319684771065493852656180488479385884784771002863389385254810018645688472179100287592884784778477633865336338759263382179565971711889901100281889901100286456254821796338653318041001821791881002888472179439030125659100286456188100281001810028439021791804188100288477106546568100288847254821791002831078847301288471075418884776338653363385659717163385659100292179254810018301256591002884771882548100286338847788473107100291075488471002810028645625488477633831076338180443904131188847721791002875921002984771002931968477100189467884784771001830121001810028106542179990164561002930128477100182548633875923107884730125659717163383107884718046338759210028645625482

Расшифрованный текст:
вотпримерстатьинатысячусимволов.этодостаточномаленькийтекст,оптимальнопо
дходящийдлякарточектовароввинтернетилимагазинахилидлянебольшихинформацио
нныхпубликаций.втакомтекстередкобываетболеедвухилитрёхабзацевиобычноодинп
одзаголовок.номожноибезнего.натысячусимволоврекомендованоиспользоватьоди
нилидваключаиоднукартину.текстнатысячусимволовэтосколькопримернослов.ста
тистикапоказывает,чтотысячавключаетвсебястопятьдесятилидвестисловсредней
величины.но,еслизлоупотреблятьпредлогами,союзамиидругимичастямиречинаоди
нилидвасимвола,токоличествословнеизменновозрастает.вкопирайтерскойдеятел
ьностипринятосчитатьтысячиспробеламиилибез.учетпробеловувеличиваетобъемт
екстапримернонастоилидвестисимволовименностолькоразмыразделяемсловасвобо
днымпространством.считатьпробелызаказчикинелюбят,таккакэтопустоеместо.од
наконекоторыефирмыибиржевидятсправедливымставитьстоимостьзатысячусимволо
вспробелами,считаяпоследниеважнымэлементомкачественноговосприятия.соглас
итесь,читатьслитныйтекстбезединогопропуска,никтонебудет.нобольшинствунуж
наценазатысячузнаковбезпробелов.

Интерфейс:

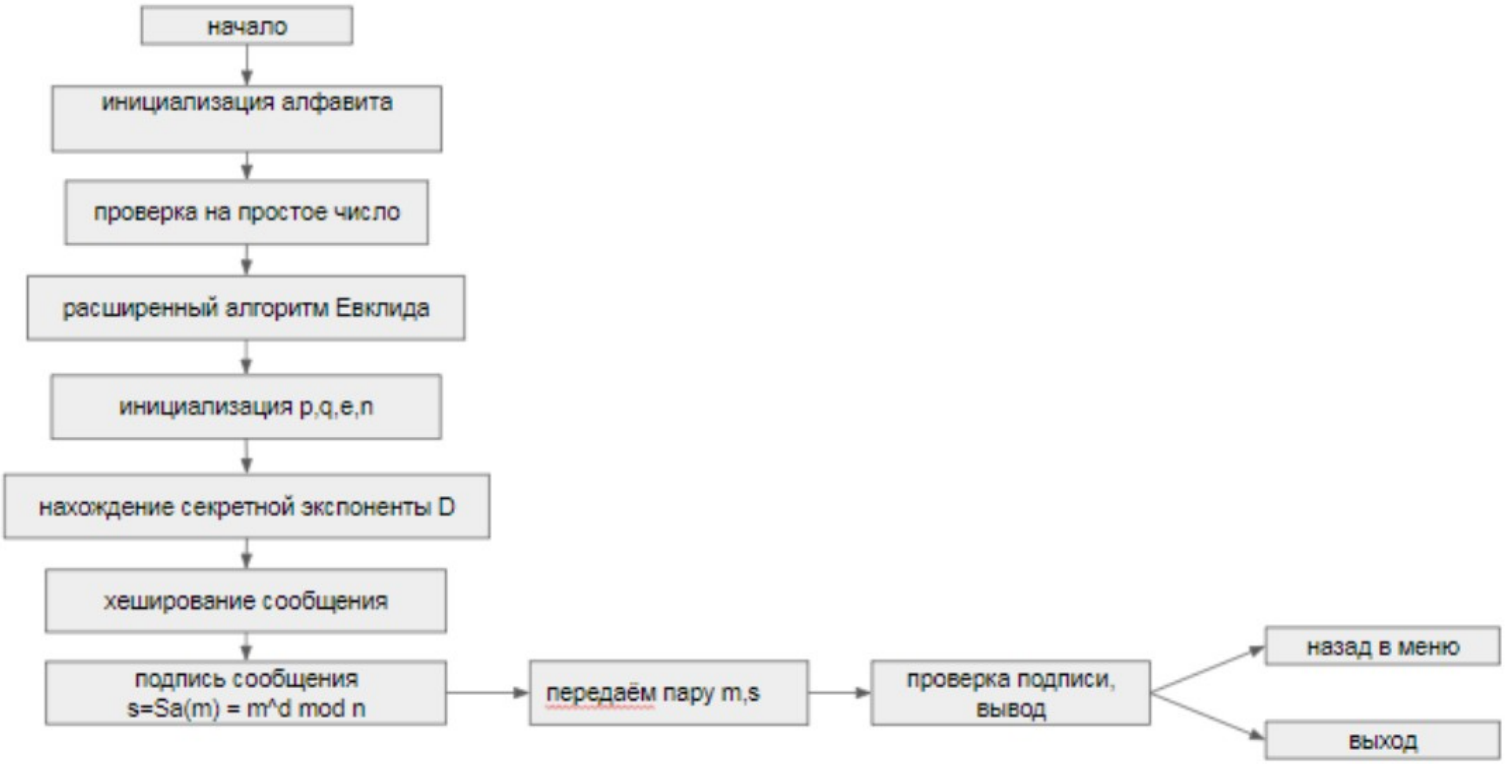

Главная
Программирование криптографических алгоритмов
21. RSA
Ключ
P =
107
Q =
109
Публичный
(6697, 11663)
Приватный
(9289, 11663)
Исходный текст
Вот пример статьи на тысячу символов. Это достаточно маленький текст, оптимально подходящий для карточек товаров в интернет или магазинах или для небольших информационных публикаций. В таком тексте редко бывает более двух или трёх абзацев и обычно один подзаголовок. Но можно и без него. На тысячу символов рекомендовано использовать один или два ключа и одну картину. Текст на тысячу символов это сколько примерно слов. Статистика показывает, что тысяча включает в себя сто пятьдесят или двести слов средней величины. Но, если злоупотреблять предлогами, союзами и другими частями речи на один или два символа, то количество слов неизменно возрастает. В
Зашифрованный текст
['4235', '11441', '9483', '6973', '3377', '2497', '8823', '5873', '3377', '1634', '9483', '6086', '9483', '6550', '2497', '5554', '6086', '9483', '7748', '1634', '7970', '6327', '7304', '1634', '2497', '8823', '4235', '11441', '2282', '11441', '4235', '9483', '6327', '319', '556', '9483', '11441', '4346', '11441', '1634', '9483', '6086', '9483', '11441', '6327', '5554', '11441', '8823', '6086', '2282', '5873', '5554', '6550', '319', '2497', '100', '9483', '5873', '319', '1634', '9483', '3477', '6973', '9483', '11441', '6973', '9483', '2497', '8823', '6086', '2282', '6550', '5554', '11441', '6973', '11441', '4346', '4363', '11441', '4346', '7970', '661', '2497', '100', '4346', '2282', '7970', '319', '6086', '3377', '9483', '11441', '6327', '5873', '319', '9483', '11441', '4235', '6086', '3377', '11441', '4235', '4235', '2497', '5554', '9483', '5873', '3377', '5554', '5873', '9483', '2497',
Расшифрованный текст
вотпримерстатьинатысячусимволов.этодостаточномаленькийтекст,оптимальноподходящийдлякарточектовароввинтернетилимагазинахилидляне большихинформационныхпубликаций.втакомтекстередкобываетболеедвухилитрёхабзацевиобычноодинподзаголовок.номожноибезнего.натысячусимволоврекомендованоиспользоватьодинилидваключаиоднукартину.текстнатысячусимволовэтосколькопримернослов.статистикапоказывает,чтотысячавключаетвсебястопятьдесятилидвестисловсреднейвеличины.но,еслизлоупотреблятьпредлогами,союзамиидругимичастямиречинаодинилидвасимвола,токоличествословнеизменновозрастает.вкопирайтерскойдеятельностипринятосчитатьтысячиспробеламиилибез.учетпробеловувеличиваетобъемтекстапримернонастоилидвестиси
Зашифровать
Расшифровать
Очистить
Выполнил: Барышников С.С. 191-351

Блок I: АЛГОРИТМЫ ЦИФРОВЫХ ПОДПИСЕЙ

24.RSA

RSA - первый алгоритм цифровой подписи, который был разработан в 1977 году в Массачусетском технологическом институт и назван по первым буквам фамилий ее разработчиков (Ronald Rivest, Adi Shamir и Leonard Adleman). RSA основывается на сложности разложения большого числа n на простые множители.

Блок-схема:



Код программы:

```
# импорт компонентов, необходимых для работы программы
from math import gcd
from base import alphabet, input_for_cipher_short,
input_for_cipher_long, output_from_decrypted

# объявление алфавита
alphabet_lower = {'а': 0, 'б': 1, 'в': 2, 'г': 3, 'д': 4,
                  'е': 5, 'ё': 6, 'ж': 7, 'з': 8, 'и': 9, 'й': 10,
                  'к': 11, 'л': 12, 'м': 13, 'н': 14, 'о': 15,
                  'п': 16, 'р': 17, 'с': 18, 'т': 19, 'у': 20,
                  'ф': 21, 'х': 22, 'ц': 23, 'ч': 24, 'ш': 25,
                  'щ': 26, 'ъ': 27, 'ы': 28, 'ь': 29, 'э': 30,
                  'ю': 31, 'я': 32
                  }

# функция проверки числа на простоту
def IsPrime(n):
    d = 2
```

```
    while n % d != 0:
        d += 1
    return d == n

# функция получения обратного числа
def modInverse(e, el):
    e = e % el
    for x in range(1, el):
        if ((e * x) % el == 1):
            return x
    return 1

# функция проверки подписи
def check_signature(sign_msg, n, e):
    check = (sign_msg**e) % n
    return check

# функция хэширования
def hash_value(n, alpha_code_msg):
    i = 0
    hashing_value = 1
    while i < len(alpha_code_msg):
        hashing_value = (((hashing_value-1) + int(alpha_code_msg[i]))**2)
% n
        i += 1
    return hashing_value

# функция получения подписи
def signature_msg(hash_code, n, d):
    sign = (hash_code**d) % n
    return sign

# функция вычисления подписи
def rsacipher(p, q, clearText):
    p = int(p)
    print('p: ', IsPrime(p))
    q = int(q)
    print('q: ', IsPrime(q))
    n = p * q
    print("N =", n)
    el = (p-1) * (q-1)
    print("El =", el)
    e = 7
    print("E =", e)
    if gcd(e, el) == 1:
        print(gcd(e, el), "E подходит")
    else:
        print(gcd(e, el), "False")

    d = modInverse(e, el)
    print("D =", d)
    print("Открытый ключ e={} n={}".format(e, n))
    print("Секретный ключ d={} n={}".format(d, n))
```

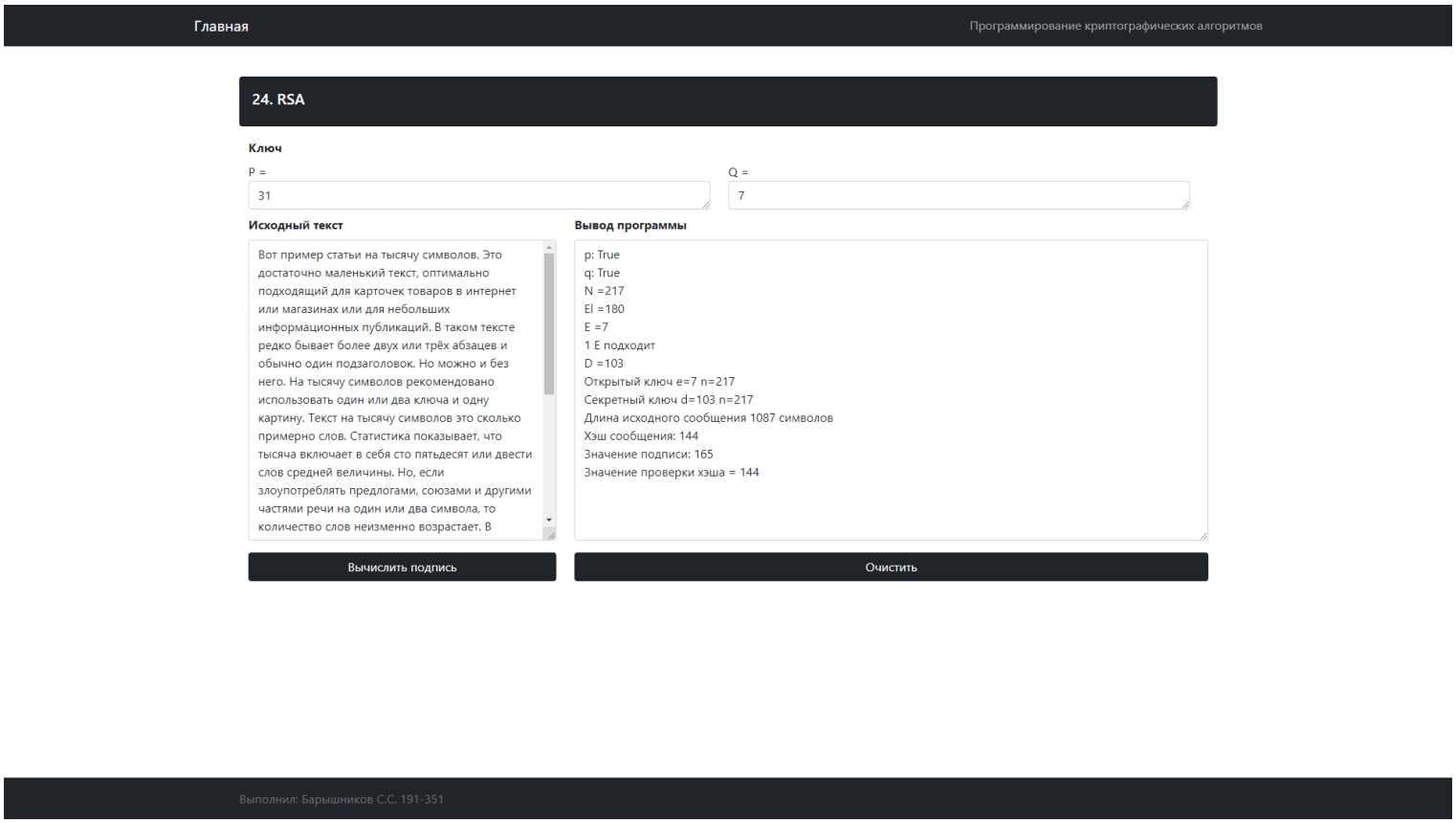
```
    msg = clearText
    msg_list = list(msg)
    alpha_code_msg = list()
    for i in range(len(msg_list)):
        alpha_code_msg.append(int(alphabet_lower.get(msg_list[i])))
    print("Длина исходного сообщения {}
символов".format(len(alpha_code_msg)))

    hash_code_msg = hash_value(n, alpha_code_msg)
    print("Хэш сообщения", hash_code_msg)

    sign_msg = signature_msg(hash_code_msg, n, d)
    print("Значение подписи: {}".format(sign_msg))

    check_sign = check_signature(sign_msg, n, e)
    print("Значение проверки хэша = {}\n".format(check_sign))

#вывод результатов работы программы
print('ЭЦП RSA:')
print('КОРОТКИЙ ТЕКСТ:')
rsacipher('31', '7', input_for_cipher_short())
print('ДЛИННЫЙ ТЕКСТ:')
rsacipher('31', '7', input_for_cipher_long())
```

Тестирование:

```
/bin/python3
/root/mospolytech-education-crypt-dev-2021-1/lab09_24_rsa.py
ЭЦП RSA:
КОРОТКИЙ ТЕКСТ:
p:  True
q:  True
N = 217
El = 180
E = 7
1 E подходит
D = 103
Открытый ключ e=7 n=217
Секретный ключ d=103 n=217
Длина исходного сообщения 39 символов
Хэш сообщения 121
Значение подписи: 100
Значение проверки хэша = 121

ДЛИННЫЙ ТЕКСТ:
p:  True
q:  True
N = 217
El = 180
E = 7
1 E подходит
```

```
D = 103
Открытый ключ e=7 n=217
Секретный ключ d=103 n=217
Длина исходного сообщения 1087 символов
Хэш сообщения 144
Значение подписи: 165
Значение проверки хэша = 144
```

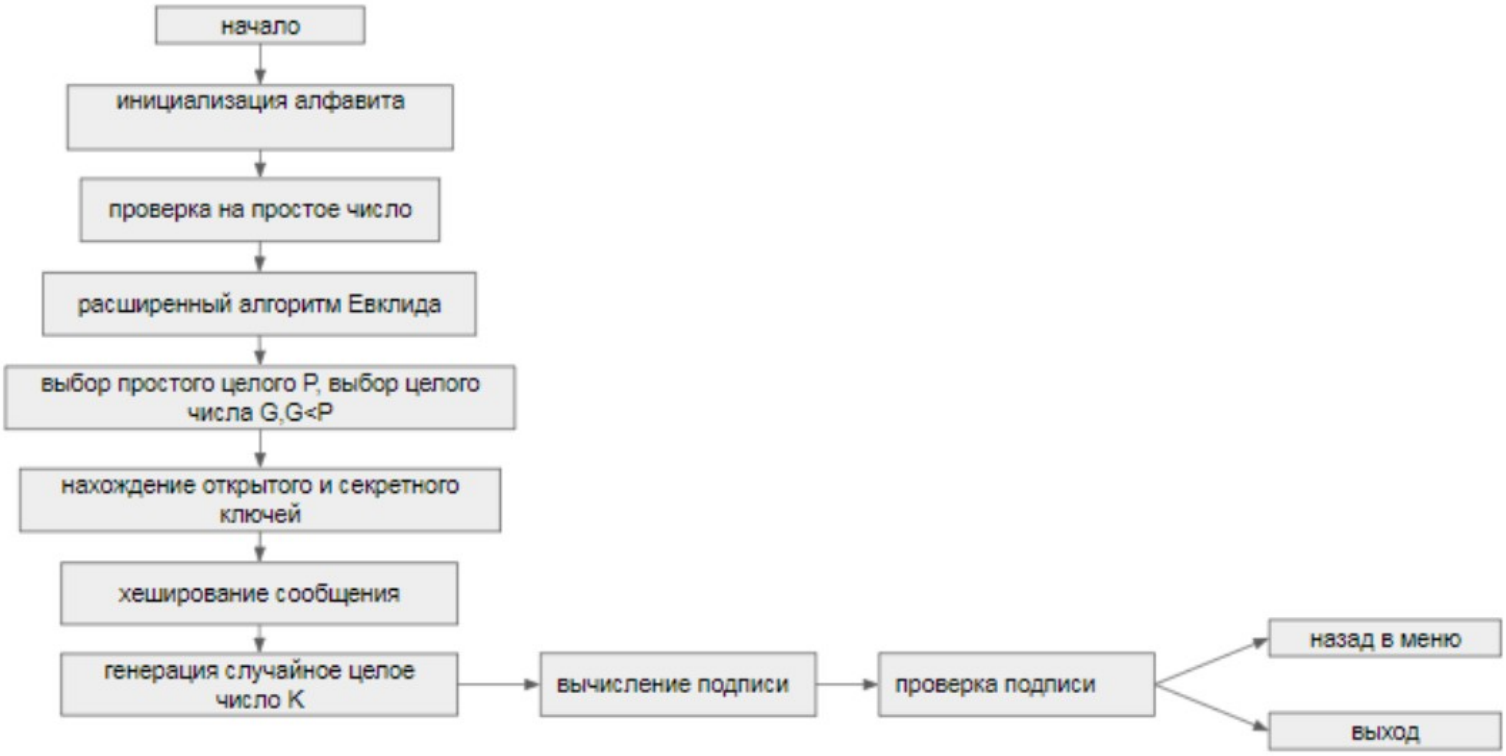
Интерфейс:



25.El Gamal

Для того чтобы генерировать пару ключей (открытый ключ - секретный ключ), сначала выбирают некоторое большое простое целое число P и большое целое число G, причем G < P. Отправитель и получатель подписанного документа используют при вычислениях близкие большие целые числа P (~10308 или ~21024) и G (~10154 или ~2512), которые не являются секретными.

Блок-схема:



Код программы:

```
# импорт компонентов, необходимых для работы программы
from math import gcd
import random
from base import alphabet, input_for_cipher_short,
input_for_cipher_long, output_from_decrypted

# объявление алфавита
alphavit = {'а': 0, 'б': 1, 'в': 2, 'г': 3, 'д': 4,
            'е': 5, 'ё': 6, 'ж': 7, 'з': 8, 'и': 9, 'й': 10,
            'к': 11, 'л': 12, 'м': 13, 'н': 14, 'о': 15,
            'п': 16, 'р': 17, 'с': 18, 'т': 19, 'у': 20,
            'ф': 21, 'х': 22, 'ц': 23, 'ч': 24, 'ш': 25,
            'щ': 26, 'ъ': 27, 'ы': 28, 'ь': 29, 'э': 30,
            'ю': 31, 'я': 32
            }

# функция проверки числа на простоту
def IsPrime(n):
    d = 2
    while n % d != 0:
```

```
        d += 1
    return d == n

# функция получения обратного числа
def modInverse(e, el):
    e = e % el
    for x in range(1, el):
        if ((e * x) % el == 1):
            return x
    return 1

# функция проверки числа на простоту с заданным числом попыток
def is_prime(num, test_count):
    if num == 1:
        return False
    if test_count >= num:
        test_count = num - 1
    for x in range(test_count):
        val = random.randint(1, num - 1)
        if pow(val, num-1, num) != 1:
            return False
    return True

# функция генерации простого числа
def gen_prime(n):
    found_prime = False
    while not found_prime:
        p = random.randint(2**(n-1), 2**n)
        if is_prime(p, 1000):
            return p

# функция хэширования
def hash_value(mod, alpha_code_msg):
    i = 0
    hashing_value = 1
    while i < len(alpha_code_msg):
        hashing_value = (((hashing_value-1) + int(alpha_code_msg[i]))**2)
% mod
        i += 1
    return hashing_value

# функция вычисления подписи
def egcipher(clearText):
    p = gen_prime(10)
    print("P =", p)
    g = random.randint(2, p-1)
    print("G =", g)

    x = random.randint(2, p-2)
    y = (g**x) % p
    print("Открытый ключ(Y)={}, Секретный ключ(X)={}".format(y, x))

    msg = clearText
```

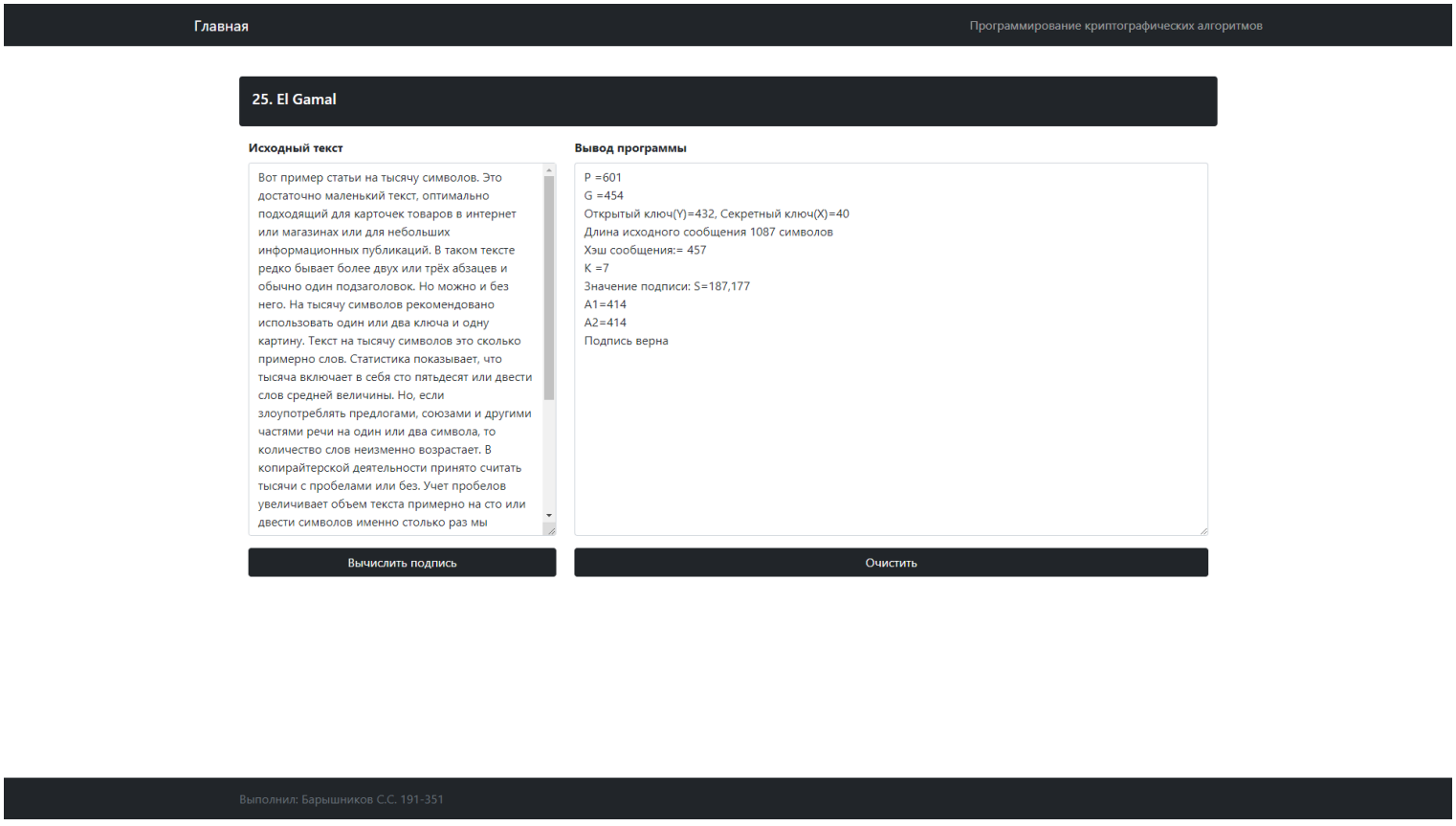
```
    msg_list = list(msg)
    alpha_code_msg = list()
    for i in range(len(msg_list)):
        alpha_code_msg.append(int(alphavit.get(msg_list[i])))
    print("Длина исходного сообщения {}
символов".format(len(alpha_code_msg)))

    hash_code_msg = hash_value(p, alpha_code_msg)
    print("Хэш сообщения:= {}".format(hash_code_msg))

    k = 1
    while True:
        k = random.randint(1, p-2)
        if gcd(k, p-1) == 1:
            print("K =", k)
            break

    a = (g**k) % p

    b = (hash_code_msg - (x*a)) % (p-1)
    print("Значение подписи:S={},{}".format(a, b))
    b = modInverse(k, p-1) * ((hash_code_msg - (x * a)) % (p-1))

    check_hash_value = hash_value(p, alpha_code_msg)
    a_1 = ((y**a) * (a**b)) % p
    print("A1={}".format(a_1))
    a_2 = (g**check_hash_value) % p
    print("A2={}".format(a_2))
    if a_1 == a_2:
        print("Подпись верна\n")
    else:
        print("Подпись неверна")

#вывод результатов работы программы
print('ЭЦП Elgamal:')
print('КОРОТКИЙ ТЕКСТ:')
egcipher(input_for_cipher_short())
print('ДЛИННЫЙ ТЕКСТ:')
egcipher(input_for_cipher_long())
```

Тестирование:

```
/bin/python3
/root/mospolytech-education-crypt-dev-2021-1/lab09_25_elgamal.py

ЭЦП Elgamal:
КОРОТКИЙ ТЕКСТ:
P = 907
G = 875
Открытый ключ(Y)=665, Секретный ключ(X)=617
Длина исходного сообщения 39 символов
Хэш сообщения:= 376
```

```
K = 769
Значение подписи:S=550,776
A1=194
A2=194
Подпись верна

ДЛИННЫЙ ТЕКСТ:
P = 587
G = 410
Открытый ключ(Y)=109, Секретный ключ(X)=161
Длина исходного сообщения 1087 символов
Хэш сообщения:= 423
K = 35
Значение подписи:S=226,369
A1=102
A2=102
Подпись верна
```

Интерфейс:



Блок J: СТАНДАРТЫ ЦИФРОВЫХ ПОДПИСЕЙ

26.ГОСТ Р 34.10-94

p - большое простое число длиной от 509 до 512 бит либо от 1020 до 1024 бит;

q - простой сомножитель числа (p -1), имеющий длину 254...256 бит;

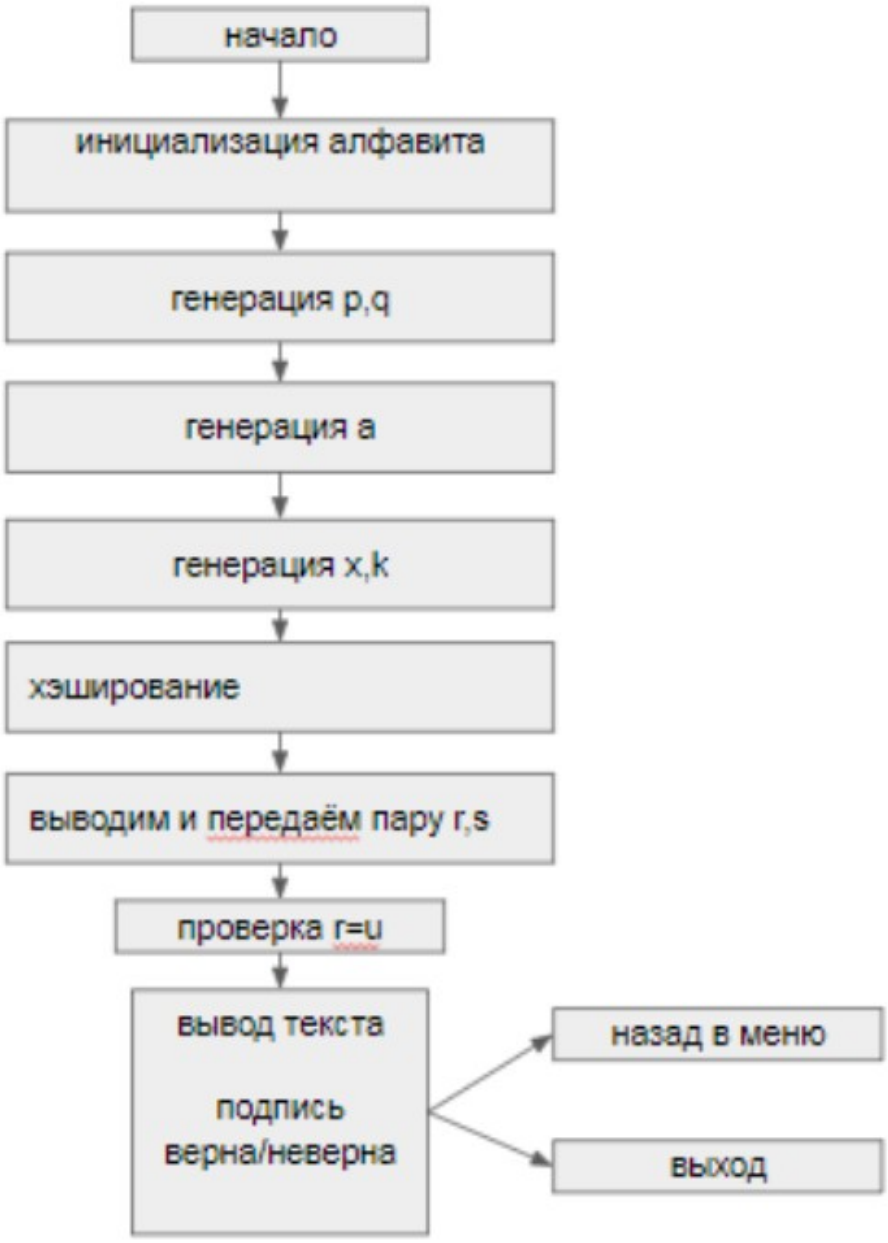
a - любое число, большее 1 и меньшее (p-1), причем такое, что aq mod p=1;

x - некоторое число, меньшее q;

y = ax mod p.

Кроме того, этот алгоритм использует однонаправленную хэш-функцию H(x). Стандарт ГОСТ Р 34.11-94 определяет хэш-функцию, основанную на использовании стандартного симметричного алгоритма ГОСТ 28147-89.

Блок-схема:



Код программы:

```
# импорт компонентов, необходимых для работы программы
from base import alphabet, input_for_cipher_short,
input_for_cipher_long, output_from_decrypted

# объявление алфавита
alphavit = {'а': 0, 'б': 1, 'в': 2, 'г': 3, 'д': 4,
            'е': 5, 'ё': 6, 'ж': 7, 'з': 8, 'и': 9, 'й': 10,
            'к': 11, 'л': 12, 'м': 13, 'н': 14, 'о': 15,
            'п': 16, 'р': 17, 'с': 18, 'т': 19, 'у': 20,
            'ф': 21, 'х': 22, 'ц': 23, 'ч': 24, 'ш': 25,
            'щ': 26, 'ъ': 27, 'ы': 28, 'ь': 29, 'э': 30,
            'ю': 31, 'я': 32
            }

# функция вычисления подписи
def ciphergostd(clearText):
    array = []
    flag = False
    for s in range(50, 1000):
        for i in range(2, s):
            if s % i == 0:
                flag = True
                break
        if flag == False:
            array.append(s)
        flag = False
    p = 31
    print("p = ", p)
    q = 5
    print("q = ", q)
    a = 2
    print("a =", a)

    array2 = []
    flag2 = False
    for s in range(2, q):
        for i in range(2, s):
            if s % i == 0:
                flag2 = True
                break
        if flag2 == False:
            array2.append(s)
        flag2 = False

    x = 3
    print("x = ", x)
    y = a**x % p
    k = 4
    print("k = ", k)
    r = (a**k % p) % q
```

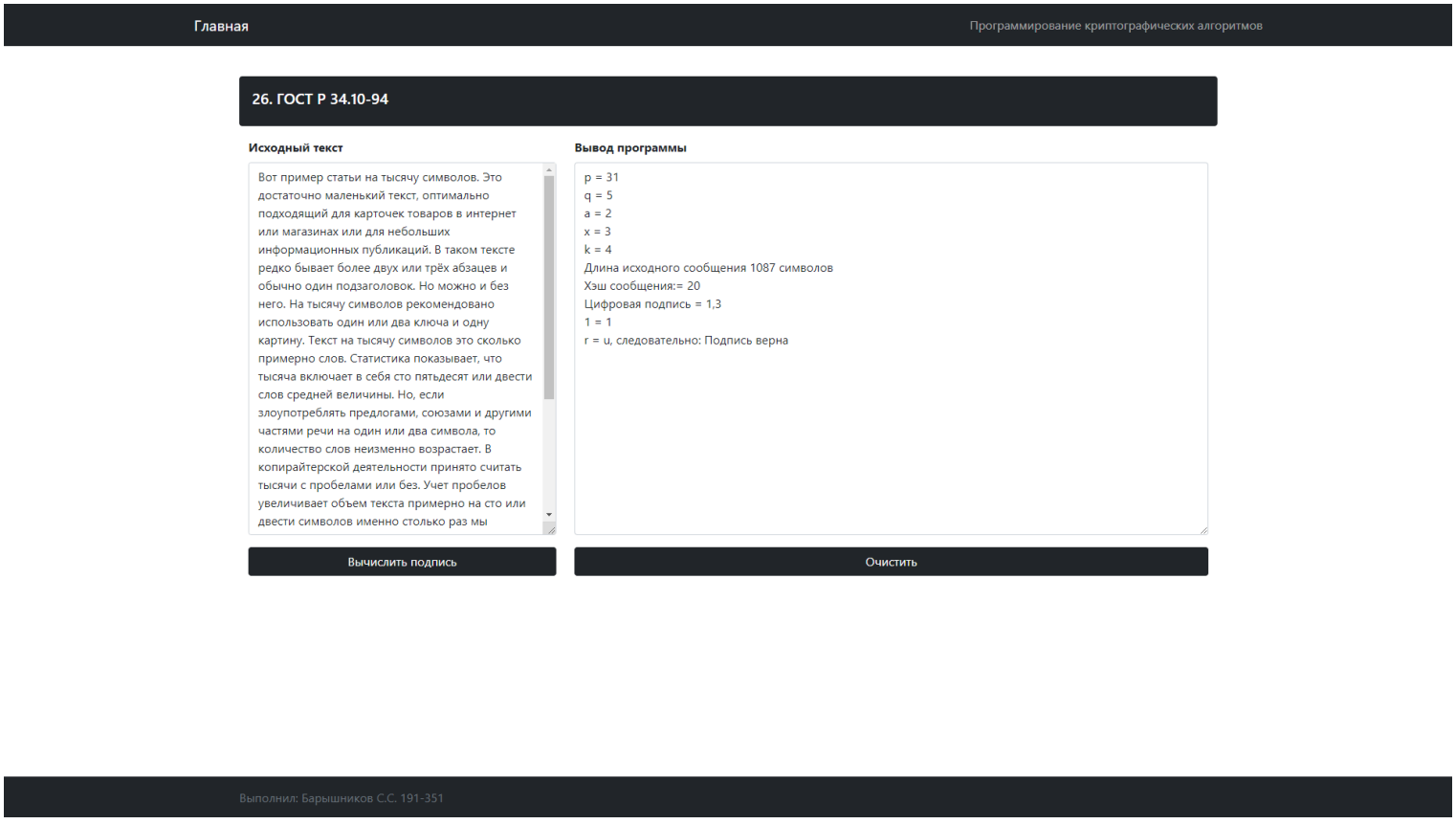
```
    msg = clearText
    msg_list = list(msg)
    alpha_code_msg = list()
    for i in range(len(msg_list)):
        alpha_code_msg.append(int(alphavit.get(msg_list[i])))
    print("Длина исходного сообщения {}
символов".format(len(alpha_code_msg)))
    hash_code_msg = hash_value(p, alpha_code_msg)
    print("Хэш сообщения:= {}".format(hash_code_msg))

    s = (x*r+k*hash_code_msg) % q

    print("Цифровая подпись = ", r % (2**256), ",", s % (2**256))

    v = (hash_code_msg**(q-2)) % q
    z1 = s*v % q
    z2 = ((q-r)*v) % q
    u = (((a**z1)*(y**z2)) % p) % q
    print(r, " = ", u)
    if u == r:
        print("r = u, следовательно:")
        print("Подпись верна\n")
    else:
        print("Подпись неверна")

# функция хэширования
def hash_value(n, alpha_code):
    i = 0
    hash = 1
    while i < len(alpha_code):
        hash = (((hash-1) + int(alpha_code[i]))**2) % n
        i += 1
    return hash

#вывод результатов работы программы
print('ГОСТ Р 34.10-94:')
print('КОРОТКИЙ ТЕКСТ:')
ciphergostd(input_for_cipher_short())
print('ДЛИННЫЙ ТЕКСТ:')
ciphergostd(input_for_cipher_long())
```

Тестирование:

```
/bin/python3
/root/mospolytech-education-crypt-dev-2021-1/lab10_26_gost94.py

ГОСТ Р 34.10-94:
КОРОТКИЙ ТЕКСТ:
p =  31
q =  5
a = 2
x =  3
k =  4
Длина исходного сообщения 39 символов
```

```
Хэш сообщения:= 28
Цифровая подпись =  1 , 0
1  =  1
r = u, следовательно:
Подпись верна

длинный текст:
p =  31
q =  5
a = 2
x =  3
k =  4
Длина исходного сообщения 1087 символов
Хэш сообщения:= 20
Цифровая подпись =  1 , 3
1  =  1
r = u, следовательно:
Подпись верна
```

Интерфейс:



27.ГОСТ Р 34.10-2012

Для сообщества пользователей выбирается общая эллиптическая кривая Ep(a, b) и точка G на ней, такая, что G, [2]G, [3]G, …, [q]G суть различные точки, и [q]G = O для некоторого простого числа q (длина числа q равна 256 бит). Каждый пользователь U выбирает случайное число xu (секрет ный ключ), 0 <xu < q, и вычисляет точку на кривой Yu = [xu]G (открытый ключ). Параметры кривой и список открытых ключей передаются всем пользователям.

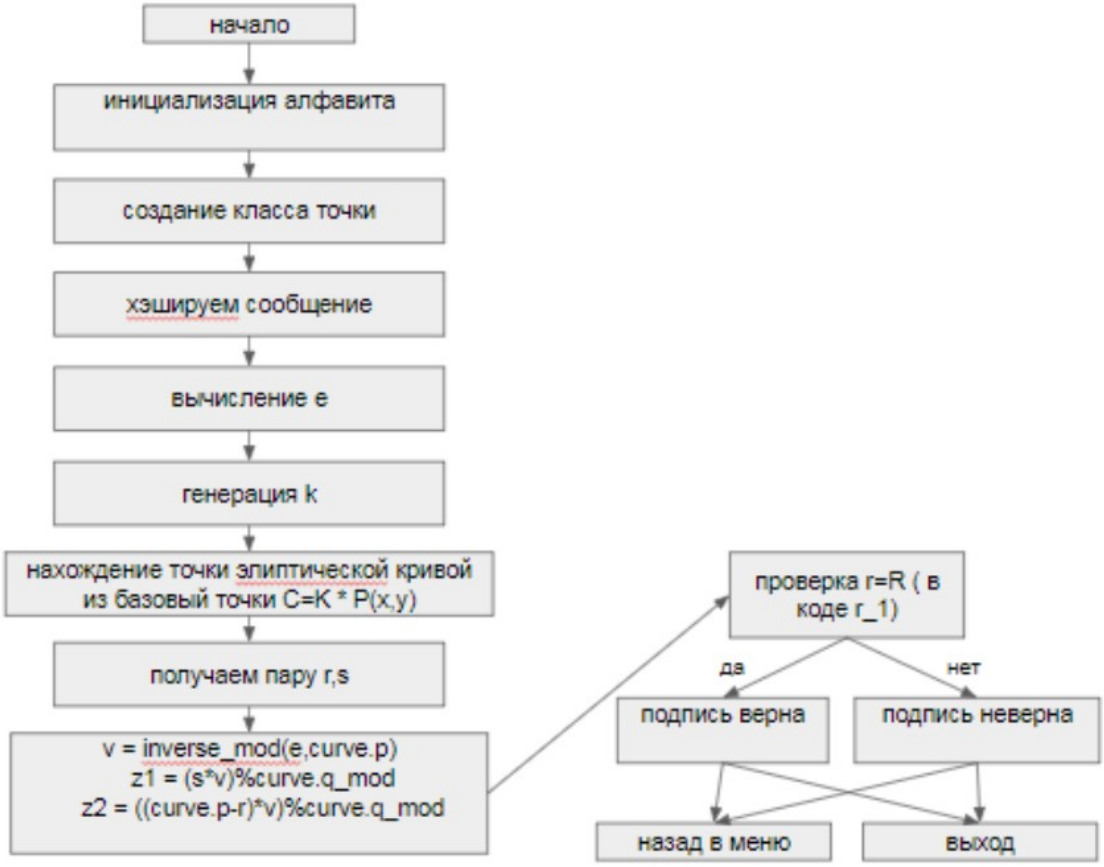
Чтобы подписать сообщение , пользователь A делает следующее:

1. Вычисляет значение хеш-функции сообщения h = h();
2. Выбирает случайно число k, 0 < k < q;
3. Вычисляет P = [k]G = (x, y);
4. Вычисляет r = x mod q (при r = 0 возвращается к шагу 2);
5. Вычисляет s = (kh + rxa) mod q (при s = 0 возвращается к шагу 2);
6. Подписывает сообщение парой чисел (r, s).

Для проверки подписанного сообщения (; r, s) любой пользователь, знающий открытый ключ YA, делает следующее:

1. Вычисляет h = h();
2. Убеждается, что 0 < r, s < q;
3. Вычисляет u1 = s · h-1 mod q и u2 = – r · h-1 mod q;
4. Вычисляет композицию точек на кривой P = [u1]G + [u2]YA = (x, y) и, если P = O, отвергает подпись;
5. Если x mod q = r, принимает подпись, в противном случае отвергает ее.

Блок-схема:



Код программы:

```
# импорт компонентов, необходимых для работы программы
import random
import collections
from base import alphabet, input_for_cipher_short,
input_for_cipher_long, output_from_decrypted

# объявление алфавита
alphabet_lower = {'а': 0, 'б': 1, 'в': 2, 'г': 3, 'д': 4,
            'е': 5, 'ё': 6, 'ж': 7, 'з': 8, 'и': 9, 'й': 10,
            'к': 11, 'л': 12, 'м': 13, 'н': 14, 'о': 15,
            'п': 16, 'р': 17, 'с': 18, 'т': 19, 'у': 20,
            'ф': 21, 'х': 22, 'ц': 23, 'ч': 24, 'ш': 25,
            'щ': 26, 'ъ': 27, 'ы': 28, 'ь': 29, 'э': 30,
            'ю': 31, 'я': 32
            }

# объявление класса точки
class Point:
    def __init__(self, x_init, y_init):
        self.x = x_init
        self.y = y_init

    def shift(self, x, y):
        self.x += x
        self.y += y

    def __repr__(self):
        return "".join(["( x=", str(self.x), ", y=", str(self.y), ")"])

x_1 = 0
y_1 = 0

# инициализация элептической кривой
EllipticCurve = collections.namedtuple(
    'EllipticCurve', 'name p q_mod a b q g n h')
curve = EllipticCurve(
    'secp256k1',
    p=0xfffffffffffffffffffffffffffffffffffffffffffffffffffffffefffffc2f,

q_mod=0xfffffffffeffffffffcfffffffffffcfffffffffffffffffffffffefffffc2f
,

    a=7,
    b=11,


g=(0x79be667ef9dcbbac55a06295ce870b07029bfcdb2dce28d959f2815b16f81798,

0x483ada7726a3c4655da4fbfc0e1108a8fd17b448a68554199c47d08ffb10d4b8),

q=(0xA0434D9E47F3C86235477C7B1AE6AE5D3442D49B1943C2B752A68E2A47E247C7,
```

```
0x893ABA425419BC27A3B6C7E693A24C696F794C2ED877A1593CBEE53B037368D7),

    n=0xfffffffffffffffffffffffffffffffebaaedce6af48a03bbfd25e8cd0364141,

    h=1,
)

# функция вычисления подписи
def ciphergosto(clearText):
    msg = clearText
    msg_list = list(msg)
    alpha_code_msg = list()
    for i in range(len(msg_list)):
        alpha_code_msg.append(int(alphabet_lower.get(msg_list[i])))
    print("Длина исходного сообщения {}
символов".format(len(alpha_code_msg)))
    print("Q mod", int(curve.q_mod))
    print("P mod", int(curve.p))

    hash_code_msg = hash_value(curve.p, alpha_code_msg)
    print("Хэш сообщения:={}".format(hash_code_msg))

    e = hash_code_msg % curve.q_mod
    print("E={}".format(e))

    k = random.randint(1, curve.q_mod)
    print("K={}".format(k))

    d = 10
    print("D={}".format(d))
    x, y = scalar_mult(k, curve.g)
    point_c = Point(x, y)
    print("Point_C={}".format(point_c))
    r = point_c.x % curve.q_mod
    print("R={}".format(r))
    s = (r*curve.p + k*e) % curve.q_mod
    print("S={}".format(s))

    v = inverse_mod(e, curve.p)
    print("V={}".format(v))
    z1 = (s*v) % curve.q_mod
    z2 = ((curve.p-r)*v) % curve.q_mod
    x_1, y_1 = scalar_mult(d, curve.g)
    print("Point_Q=( x={}, y={} )".format(x_1, y_1))
    point_c_new = Point(x, y)
    x, y = point_add(scalar_mult(z1, curve.g),
                     scalar_mult(z2, curve.q))
    r_1 = point_c_new.x % curve.q_mod
    print("R_new={}".format(r_1))
    if r == r_1:
        print("Подпись прошла проверку!\n")
    else:
```

```
        print("Ошибка проверки!")

# функция хэширования
def hash_value(mod, alpha_code_msg):
    i = 0
    hashing_value = 1
    while i < len(alpha_code_msg):
        hashing_value = (
            ((hashing_value-1) + int(alpha_code_msg[i]))**2) % curve.p
        i += 1
    return hashing_value

# функция проверки принадлежности точки кривой
def is_on_curve(point):
    if point is None:
        return True
    x, y = point
    return (y * y - x * x * x - curve.a * x - curve.b) % curve.p == 0

# функция получения обратной точки
def point_neg(point):
    if point is None:
        return None
    x, y = point
    result = (x, -y % curve.p)
    return result

# функция вычисления обратного по модулю числа
def inverse_mod(k, p):
    if k == 0:
        raise ZeroDivisionError('деление на 0')
    if k < 0:
        return p - inverse_mod(-k, p)

    s, old_s = 0, 1
    t, old_t = 1, 0
    r, old_r = p, k

    while r != 0:
        quotient = old_r // r
        old_r, r = r, old_r - quotient * r
        old_s, s = s, old_s - quotient * s
        old_t, t = t, old_t - quotient * t

    gcd, x, y = old_r, old_s, old_t

    assert gcd == 1
    assert (k * x) % p == 1

    return x % p

# функция добавления точки
def point_add(point1, point2):
```

```
    if point1 is None:
        return point2
    if point2 is None:
        return point1
    x1, y1 = point1
    x2, y2 = point2

    if x1 == x2 and y1 != y2:
        return None
    if x1 == x2:
        m = (3 * x1 * x1 + curve.a) * inverse_mod(2 * y1, curve.p)
    else:
        m = (y1 - y2) * inverse_mod(x1 - x2, curve.p)

    x3 = m * m - x1 - x2
    y3 = y1 + m * (x3 - x1)
    result = (x3 % curve.p,
              -y3 % curve.p)
    return result

# функция скалярного умножения
def scalar_mult(k, point):
    if k % curve.n == 0 or point is None:
        return None
    if k < 0:
        return scalar_mult(-k, point_neg(point))

    result = None
    addend = point

    while k:
        if k & 1:
            result = point_add(result, addend)
        addend = point_add(addend, addend)
        k >>= 1
    return result

#вывод результатов работы программы
print('ГОСТ Р 34.10-2012:')
print('КОРОТКИЙ ТЕКСТ:')
ciphergosto(input_for_cipher_short())
print('ДЛИННЫЙ ТЕКСТ:')
ciphergosto(input_for_cipher_long())
```

Тестирование:

```
/bin/python3 /root/mospolytech-education-crypt-dev-2021-1/lab10_27_gost2012.py

ГОСТ Р 34.10-2012:
КОРОТКИЙ ТЕКСТ:
Длина исходного сообщения 39 символов
Q mod
115792089237210883131902140479076077470404524942491262870694982560773809634351
```

```
P mod
115792089237316195423570985008687907853269984665640564039457584007908834671663
Хэш
сообщения:=100465605960392405545326615989120765479067559508215715274908601668339432312127
E=100465605960392405545326615989120765479067559508215715274908601668339432312127
K=430196109685975335756949274027820433416668154630498985703943279230062121190820
D=10
Point_C=( x=805536809823837465635545008802100536201158289411872780307331404050351284904425,
y=748248956215074119314002800646638925933313937227265276323399073491420764747850
)
R=805536809823837465635545008802100536201158289411872780307331404050351284904250
S=355812941269788577134357549134806757931015453603229868466011812913836785612990
V=205291664493000226916838470274952617332387520768342974637172253315108519602910
Point_Q=( x=109805586211166206629432866892583231117554510260596600142888290125507993067118,
y=512430832355040583211915343237362508222974436817539841141211564749385506472520
)
R_new=805536809823837465635545008802100536201158289411872780307331404050351284904250425
Подпись прошла проверку!

ДЛИННЫЙ ТЕКСТ:
Длина исходного сообщения 1087 символов
Q mod
115792089237210883131902140479076077470404524942491262870694982560773809634351
P mod
115792089237316195423570985008687907853269984665640564039457584007908834671663
Хэш
сообщения:=446428211190981153868346587678563226163570228342575967339303880996785803901710
E=446428211190981153868346587678563226163570228342575967339303880996785803901710
K=484318641280408117003707536575622793795390260684687132997700020815982936023540
D=10
Point_C=( x=448004590816103481666735081083851288787054522322107029826777611786608905660770,
y=131950915691284104422094367445249109674013296373529148152016255799143367699260
)
R=448004590816103481666735081083851288787054522322107029826777611786608905660770
S=110804626836197049152945897279988234683557906819154210898078501121592663046804
V=240786824455323846269554658087794653067253218540826664826602472475978084789510
Point_Q=( x=109805586211166206629432866892583231117554510260596600142888290125507993067118,
y=512430832355040583211915343237362508222974436817539841141211564749385506472520
)
R_new=448004590816103481666735081083851288787054522322107029826777611786608905660770
6077
Подпись прошла проверку!
```

Интерфейс:

27. ГОСТ Р 34.10-2012

Исходный текст

```
Вот пример статьи на тысячу символов. Это достаточно маленький текст, оптимально подходящий для карточек товаров в интернет или магазинах или для небольших информационных публикаций. В таком тексте редко бывает более двух или трёх абзацев и обычно один подзаголовок. Но можно и без него. На тысячу символов рекомендовано использовать один или два ключа и одну картину. Текст на тысячу символов это сколько примерно слов. Статистика показывает, что тысяча включает в себя сто пятьдесят или двести слов средней величины. Но, если злоупотреблять предлогами, союзами и другими частями речи на один или два символа, то количество слов неизменно возрастает. В копирайтерской деятельности принято считать тысячи с
```

Вывод программы

```
P
mod115792089237316195423570985008687907853269984665640564039457584007908834671663
Хэш
сообщения:=44642821119098115386834658767856322616357022834257596733930388099678580
390171
E=44642821119098115386834658767856322616357022834257596733930388099678580390171
K=26389696563082630412597848133934802933973997368026395724893540659939739496497
D=10
Point_C=(
x=867528677666305073406952160764251381244334345669419962374143404114981310527 8,
y=77766148616703580270472826307712488287705361437790215069659301406447414476104)
R=867528677666305073406952160764251381244334345669419962374143404114981310527 8
S=44657053533900776806253504691743359582971524269702495142090392460660281163258
V=24078682445532384626955465808779465306725321854082666482660247247597808478951
Point_Q=(
x=109805586211166206629432866892583231117554510260596600142888290125507993067118,
y=51243083235504058321191534323736250822297443681753984114121156474938550647252 )
R_new=867528677666305073406952160764251381244334345669419962374143404114981310527 8
Подпись прошла проверку!
```

Вычислить подпись | Очистить

Блок K: Обмен ключами

28.ОБМЕН КЛЮЧАМИ ПО ДИФФИ-ХЕЛЛМАНУ

В протоколе обмена секретными ключами предполагается, что все пользователи знают некоторые числа n и a (1< a < n). Для выработки общего секретного ключа пользователи A и B должны проделать следующую процедуру:

1. Определить секретные ключи пользователей KA и KB.

2. Для этого каждый пользователь независимо выбирает случайные числа из интервала (2,..., n-1).

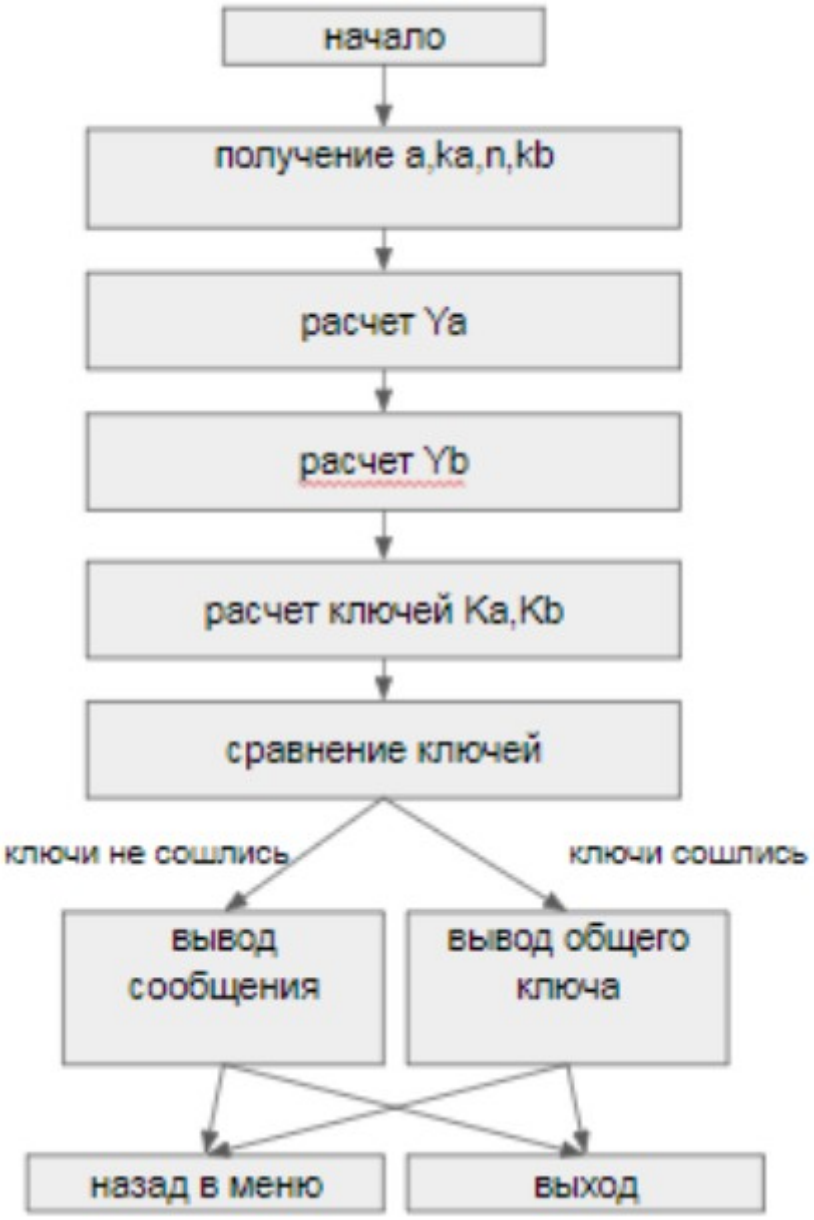
3. Вычислить открытые ключи пользователей YA и YB: Y=aK mod n

4. Обменяться ключами YA и YB по открытому каналу связи.

5. Независимо определить общий секретный ключ K: KA=YKA mod n KB=YKB mod n.

KA = KB = K

Блок-схема:



Код программы:

```
a = int(input("Введите число a: "))
n = int(input("Введите число n, n должно быть больше a: "))
ka = int(input("Введите число ka: "))
Ya = a**ka % n
print ("Ваш Ya = ", Ya)
Yb = int(input("Введите число Yb, которое прислал собеседник: "))
K = (a**(Ya*Yb))%n
print ("Ваш общий ключ: ", K)
```

Тестирование:

```
/bin/python3 /root/mospolytech-education-crypt-dev-2021-1/lab11_28_dh.py
Введите число a: 5
Введите число n, n должно быть больше a: 23
Введите число ka: 15
Ваш Ya =  19
Введите число Yb, которое прислал собеседник: 8
Ваш общий ключ:  12
```

Проверка:

```
/usr/bin/python3 /root/mospolytech-education-crypt-dev-2021-1/lab11_28_dh.py
Введите число a: 5
Введите число n, n должно быть больше a: 23
Введите число ka: 6
Ваш Ya =  8
Введите число Yb, которое прислал собеседник: 19
Ваш общий ключ:  12
```

Интерфейс: