Алябьев В.О. А-07-18
Лабораторная работа №3

1. Создать пустую строку с именем myStringN ="" (N- номер по списку в группе) и скопировать в неё из файла plaintext.doc страницу с номером N (без рисунков). (Глава 3, стр 1)

In[1]:=
```
myString1 = "ГЛАВА 3. СОВРЕМЕННЫЕ СИММЕТРИЧНЫЕ КРИПТОСИСТЕМЫ
По мнению К. Шеннона [101], в практических шифрах необходимо
    использовать два общих принципа: рассеивание и перемешивание.

Рассеивание представляет собой распространение влияния
    одного знака открытого текста на много знаков шифртекста,
    что позволяет скрыть статистические свойства открытого текста.

Перемешивание предполагает использование таких
    шифрующих преобразований, которые усложняют восстановление
    взаимосвязи статистических свойств открытого и шифрованного
    текстов. Однако шифр должен не только затруднять
    раскрытие, но и обеспечивать легкость зашифрования и
    расшифрования при известном пользователю секретном ключе.

Распространенным способом достижения эффектов рассеивания и
    перемешивания является использование составного шифра, т.е.
    такого шифра, который может быть реализован в виде некоторой
    последовательности простых шифров, каждый из которых вносит
    свой вклад в значительное суммарное рассеивание и перемешивание.

В составных шифрах в качестве простых шифров чаще всего используются
    простые перестановки и подстановки. При перестановке просто
    перемешивают символы открытого текста, причем конкретный вид
    перемешивания определяется секретным ключом. При подстановке
    каждый символ открытого текста заменяют другим символом из того же
    алфавита, а конкретный вид подстановки также определяется секретным
    ключом. Следует заметить, что в современном блочном шифре
    блоки открытого текста и шифртекста представляют собой двоичные
    последовательности обычно длиной 64 бита. В принципе каждый блок
    может принимать 264 значений. Поэтому подстановки выполняются
    в очень большом алфавите, содержащем до 264 ⌊ 1019 символов.

При многократном чередовании простых перестановок и подстановок,
    управляемых достаточно длинным секретным ключом, можно
```

получить очень стойкий шифр с хорошим рассеиванием и перемешиванием. Рассмотренные ниже криптоалгоритмы DES, IDEA и отечественный стандарт шифрования данных построены в полном соответствии с указанной методологией.

3.1. Американский стандарт шифрования данных DES

Стандарт шифрования данных DES (Data Encryption Standard) опубликован в 1977 г. Национальным бюро стандартов США. Стандарт DES предназначен для защиты от несанкционированного доступа к важной, но несекретной информации в государственных и коммерческих организациях США. Алгоритм, положенный в основу стандарта, распространялся достаточно быстро, и уже в 1980 г. был одобрен Национальным институтом стандартов и технологий США (НИСТ). С этого момента DES превращается в стандарт не только по названию (Data Encryption Standard), но и фактически. Появляются программное обеспечение и специализированные микроЭВМ, предназначенные для шифрования и расшифрования информации в сетях передачи данных.

К настоящему времени DES является наиболее распространенным алгоритмом, используемым в системах защиты коммерческой информации. Более того, реализация алгоритма DES в таких системах становится признаком хорошего тона.

Основные достоинства алгоритма DES:

используется только один ключ длиной 56 бит;

зашифровав сообщение с помощью одного пакета программ, для расшифровки можно использовать любой другой пакет программ, соответствующий стандарту DES;

относительная простота алгоритма обеспечивает высокую скорость обработки;

достаточно высокая стойкость алгоритма.

Первоначально метод, лежащий в основе стандарта DES, был разработан фирмой IBM для своих целей и реализован в виде системы Люцифер. Система Люцифер основана на комбинировании методов подстановки и перестановки и состоит из чередующейся последовательности блоков перестановки и подстановки. В ней использовался ключ длиной 128 бит, управлявший состояниями блоков перестановки и подстановки. Система Люцифер оказалась весьма сложной для практической

реализации из-за относительно малой скорости шифрования (2190 **байт/с - программная реализация**, 96970 **байт/с - аппаратная реализация**).

Алгоритм DES **также использует комбинацию подстановок и перестановок.** DES **осуществляет шифрование** 64**-битовых блоков данных с помощью** 64**-битового ключа, в котором значащими являются** 56 **бит** (**остальные** 8 **бит - проверочные биты для контроля на четность**). **Дешифрование в** DES **является операцией, обратной шифрованию, и выполняется путем повторения операций шифрования в обратной последовательности. Обобщенная схема процесса шифрования в алгоритме** DES **показана на рис.**3.1. **Процесс шифрования заключается в начальной перестановке битов** 64**-битового блока, шестнадцати циклах шифрования и, наконец, в конечной перестановке битов."**

Out[1]= ГЛАВА 3. СОВРЕМЕННЫЕ СИММЕТРИЧНЫЕ КРИПТОСИСТЕМЫ

По мнению К. Шеннона [101], в практических шифрах необходимо использовать два общих принципа: рассеивание и перемешивание.

Рассеивание представляет собой распространение влияния одного знака открытого текста на много знаков шифртекста, что позволяет скрыть статистические свойства открытого текста.

Перемешивание предполагает использование таких шифрующих преобразований, которые усложняют восстановление взаимосвязи статистических свойств открытого и шифрованного текстов. Однако шифр должен не только затруднять раскрытие, но и обеспечивать легкость зашифрования и расшифрования при известном пользователю секретном ключе.

Распространенным способом достижения эффектов рассеивания и перемешивания является использование составного шифра, т.е. такого шифра, который может быть реализован в виде некоторой последовательности простых шифров, каждый из которых вносит свой вклад в значительное суммарное рассеивание и перемешивание.

В составных шифрах в качестве простых шифров чаще всего используются простые перестановки и подстановки. При перестановке просто перемешивают символы открытого текста, причем конкретный вид перемешивания определяется секретным ключом. При подстановке каждый символ открытого текста заменяют другим символом из того же алфавита, а конкретный вид подстановки также определяется секретным ключом. Следует заметить, что в современном блочном шифре блоки открытого текста и шифртекста представляют собой двоичные последовательности обычно длиной 64 бита. В принципе каждый блок может принимать 264 значений. Поэтому подстановки

выполняются в очень большом алфавите, содержащем до 264 ⌊ 1019 символов.

При многократном чередовании простых перестановок и подстановок, управляемых достаточно длинным секретным ключом, можно получить очень стойкий шифр с хорошим рассеиванием и перемешиванием. Рассмотренные ниже криптоалгоритмы DES, IDEA и отечественный стандарт шифрования данных построены в полном соответствии с указанной методологией.

3.1. Американский стандарт шифрования данных DES

Стандарт шифрования данных DES (Data Encryption Standard) опубликован в 1977 г. Национальным бюро стандартов США. Стандарт DES предназначен для защиты от несанкционированного доступа к важной, но несекретной информации в государственных и коммерческих организациях США. Алгоритм, положенный в основу стандарта, распространялся достаточно быстро, и уже в 1980 г. был одобрен Национальным институтом стандартов и технологий США (НИСТ). С этого момента DES превращается в стандарт не только по названию (Data Encryption Standard), но и фактически. Появляются программное обеспечение и специализированные микроЭВМ, предназначенные для шифрования и расшифрования информации в сетях передачи данных.

К настоящему времени DES является наиболее распространенным алгоритмом, используемым в системах защиты коммерческой информации. Более того, реализация алгоритма DES в таких системах становится признаком хорошего тона.

Основные достоинства алгоритма DES:

используется только один ключ длиной 56 бит;

зашифровав сообщение с помощью одного пакета программ, для расшифровки можно использовать любой другой пакет программ, соответствующий стандарту DES;

относительная простота алгоритма обеспечивает высокую скорость обработки;

достаточно высокая стойкость алгоритма.

Первоначально метод, лежащий в основе стандарта DES, был разработан фирмой IBM для своих целей и реализован в виде системы Люцифер. Система Люцифер основана на комбинировании методов подстановки и перестановки и состоит из чередующейся последовательности блоков перестановки и подстановки. В ней использовался ключ длиной 128 бит, управлявший состояниями блоков

перестановки и подстановки. Система Люцифер оказалась весьма сложной для практической реализации из-за относительно малой скорости шифрования (2190 байт/с – программная реализация, 96970 байт/с – аппаратная реализация).

Алгоритм DES также использует комбинацию подстановок и перестановок. DES осуществляет шифрование 64-битовых блоков данных с помощью 64-битового ключа, в котором значащими являются 56 бит (остальные 8 бит – проверочные биты для контроля на четность). Дешифрование в DES является операцией, обратной шифрованию, и выполняется путем повторения операций шифрования в обратной последовательности. Обобщенная схема процесса шифрования в алгоритме DES показана на рис.3.1. Процесс шифрования заключается в начальной перестановке битов 64-битового блока, шестнадцати циклах шифрования и, наконец, в конечной перестановке битов.

2. Определить число символов в строке :⌣ StringLength[].

In[2]:=
```
StringLength[myString1]
```

Out[2]= 4532

3. Получить список строчных букв русского алфавита, объединить их в строку и получить список кодов, соответствующих строчным буквам : ⌣ CharacterRange[], StringJoin[],ToCharacter-Code[].

In[3]:=
```
smallLet = CharacterRange["а", "я"]
```

Out[3]= {а, б, в, г, д, е, ж, з, и, й, к, л, м, н,
о, п, р, с, т, у, ф, х, ц, ч, ш, щ, ъ, ы, ь, э, ю, я}

In[4]:=
```
smalLetString = StringJoin[smallLet]
```

Out[4]= абвгдежзийклмнопрстуфхцчшщъыьэюя

In[5]:=
```
ToCharacterCode[smalLetString]
```

Out[5]= {1072, 1073, 1074, 1075, 1076, 1077, 1078, 1079, 1080, 1081,
1082, 1083, 1084, 1085, 1086, 1087, 1088, 1089, 1090, 1091, 1092,
1093, 1094, 1095, 1096, 1097, 1098, 1099, 1100, 1101, 1102, 1103}

4. Получить список прописных букв русского алфавита, объединить их в строку и получить список кодов, соответствующих прописным буквам.

In[6]:=
```
bigLet = CharacterRange["А", "Я"]
```

Out[6]= {А, Б, В, Г, Д, Е, Ж, З, И, Й, К, Л, М, Н,
О, П, Р, С, Т, У, Ф, Х, Ц, Ч, Ш, Щ, Ъ, Ы, Ь, Э, Ю, Я}

In[7]:=
```
bigLetString = StringJoin[bigLet]
```
Out[7]=
```
АБВГДЕЖЗИЙКЛМНОПРСТУФХЦЧШЩЪЫЬЭЮЯ
```

In[8]:=
```
ToCharacterCode[bigLetString]
```
Out[8]=
```
{1040, 1041, 1042, 1043, 1044, 1045, 1046, 1047, 1048, 1049,
 1050, 1051, 1052, 1053, 1054, 1055, 1056, 1057, 1058, 1059, 1060,
 1061, 1062, 1063, 1064, 1065, 1066, 1067, 1068, 1069, 1070, 1071}
```

5. Определить код символа "пробел". Определить коды символов ":" ,"-" ,";".

In[9]:=
```
ToCharacterCode[{" ", ":" , "-" , ";"}]
```
Out[9]=
```
{{32}, {58}, {45}, {59}}
```

6. Определить коды цифр.

In[10]:=
```
ToCharacterCode[CharacterRange["0", "9"]]
```
Out[10]=
```
{{48}, {49}, {50}, {51}, {52}, {53}, {54}, {55}, {56}, {57}}
```

7. Объединить списки прописных и строчных букв с помощью функции Riffle[] и разбить (Partition[]) объединеный список на пары: {{А,а},{Б,б},{В,в},и т.д.

In[11]:=
```
Riffle[smallLet, bigLet]
```
Out[11]=
```
{а, А, б, Б, в, В, г, Г, д, Д, е, Е, ж, Ж, з, З, и, И, й, Й,
 к, К, л, Л, м, М, н, Н, о, О, п, П, р, Р, с, С, т, Т, у, У, ф, Ф,
 х, Х, ц, Ц, ч, Ч, ш, Ш, щ, Щ, ъ, Ъ, ы, Ы, ь, Ь, э, Э, ю, Ю, я, Я}
```

In[12]:=
```
pares = Partition[Riffle[bigLet, smallLet], 2]
```
Out[12]=
```
{{А, а}, {Б, б}, {В, в}, {Г, г}, {Д, д}, {Е, е}, {Ж, ж}, {З, з}, {И, и}, {Й, й}, {К, к},
 {Л, л}, {М, м}, {Н, н}, {О, о}, {П, п}, {Р, р}, {С, с}, {Т, т}, {У, у}, {Ф, ф}, {Х, х},
 {Ц, ц}, {Ч, ч}, {Ш, ш}, {Щ, щ}, {Ъ, ъ}, {Ы, ы}, {Ь, ь}, {Э, э}, {Ю, ю}, {Я, я}}
```

8.Сформировать правило замены прописных букв русского языка на строчные с применением функции Apply[Rule,***,{1}] в виде: {А→а,Б→б,В→в, и т.д.

In[13]:=
```
Rules = Apply[Rule, pares , {1}]
```
Out[13]=
```
{А → а, Б → б, В → в, Г → г, Д → д, Е → е, Ж → ж, З → з, И → и, Й → й, К → к,
 Л → л, М → м, Н → н, О → о, П → п, Р → р, С → с, Т → т, У → у, Ф → ф, Х → х,
 Ц → ц, Ч → ч, Ш → ш, Щ → щ, Ъ → ъ, Ы → ы, Ь → ь, Э → э, Ю → ю, Я → я}
```

9. Провести замену прописных букв на строчные в строке myStringN, используя функцию StringReplace[myStringN,{"s1"->"spl","s2"->"sp2",...}].

In[14]:=
```
lowstr1 = StringReplace[myString1 , Rules ]
```

Out[14]= глава 3. современные симметричные криптосистемы

по мнению к. шеннона [101], в практических шифрах необходимо использовать два общих принципа: рассеивание и перемешивание.

рассеивание представляет собой распространение влияния одного знака открытого текста на много знаков шифртекста, что позволяет скрыть статистические свойства открытого текста.

перемешивание предполагает использование таких шифрующих преобразований, которые усложняют восстановление взаимосвязи статистических свойств открытого и шифрованного текстов. однако шифр должен не только затруднять раскрытие, но и обеспечивать легкость зашифрования и расшифрования при известном пользователю секретном ключе.

распространенным способом достижения эффектов рассеивания и перемешивания является использование составного шифра, т.е. такого шифра, который может быть реализован в виде некоторой последовательности простых шифров, каждый из которых вносит свой вклад в значительное суммарное рассеивание и перемешивание.

в составных шифрах в качестве простых шифров чаще всего используются простые перестановки и подстановки. при перестановке просто перемешивают символы открытого текста, причем конкретный вид перемешивания определяется секретным ключом. при подстановке каждый символ открытого текста заменяют другим символом из того же алфавита, а конкретный вид подстановки также определяется секретным ключом. следует заметить, что в современном блочном шифре блоки открытого текста и шифртекста представляют собой двоичные последовательности обычно длиной 64 бита. в принципе каждый блок может принимать 264 значений. поэтому подстановки выполняются в очень большом алфавите, содержащем до $\lfloor 264 \rfloor$ 1019 символов.

при многократном чередовании простых перестановок и подстановок, управляемых достаточно длинным секретным ключом, можно получить очень стойкий шифр с хорошим рассеиванием и перемешиванием. рассмотренные ниже криптоалгоритмы DES, IDEA и отечественный стандарт шифрования данных построены в полном соответствии с указанной методологией.

3.1. американский стандарт шифрования данных DES

стандарт шифрования данных DES (Data Encryption Standard) опубликован в 1977 г. национальным бюро стандартов сша. стандарт DES предназначен для защиты от несанкционированного доступа к важной, но несекретной

информации в государственных и коммерческих организациях сша. алгоритм, положенный в основу стандарта, распространялся достаточно быстро, и уже в 1980 г. был одобрен национальным институтом стандартов и технологий сша (нист). с этого момента DES превращается в стандарт не только по названию (Data Encryption Standard), но и фактически. появляются программное обеспечение и специализированные микроэвм, предназначенные для шифрования и расшифрования информации в сетях передачи данных.

к настоящему времени DES является наиболее распространенным алгоритмом, используемым в системах защиты коммерческой информации. более того, реализация алгоритма DES в таких системах становится признаком хорошего тона.

основные достоинства алгоритма DES:

используется только один ключ длиной 56 бит;

зашифровав сообщение с помощью одного пакета программ, для расшифровки можно использовать любой другой пакет программ, соответствующий стандарту DES;

относительная простота алгоритма обеспечивает высокую скорость обработки;

достаточно высокая стойкость алгоритма.

первоначально метод, лежащий в основе стандарта DES, был разработан фирмой IBM для своих целей и реализован в виде системы люцифер. система люцифер основана на комбинировании методов подстановки и перестановки и состоит из чередующейся последовательности блоков перестановки и подстановки. в ней использовался ключ длиной 128 бит, управлявший состояниями блоков перестановки и подстановки. система люцифер оказалась весьма сложной для практической реализации из-за относительно малой скорости шифрования (2190 байт/с – программная реализация, 96970 байт/с – аппаратная реализация).

алгоритм DES также использует комбинацию подстановок и перестановок. DES осуществляет шифрование 64-битовых блоков данных с помощью 64-битового ключа, в котором значащими являются 56 бит (остальные 8 бит – проверочные биты для контроля на четность). дешифрование в DES является операцией, обратной шифрованию, и выполняется путем повторения операций шифрования в обратной последовательности. обобщенная схема процесса шифрования в алгоритме DES показана на рис.3.1. процесс шифрования заключается в начальной перестановке битов 64-битового блока, шестнадцати циклах шифрования и, наконец, в конечной перестановке битов.

10. Разработать функцию пользователя для замены прописных букв на строчные русского алфавита (аналог ToLowerCase["ABCD"]), с применением операции Module[]. Входной

параметр функции пользователя – произвольная строка, выход – модифицированная строка с заменой прописных букв на строчные.

In[15]:=
```
toLowCase[str_] :=
Module[{s = str},
StringReplace[s, Rules]]
```

11. Провести замену прописных букв на строчные в строке myStringN, используя функцию пользователя и сравнить результат со строкой п. 9, определив расс тояние Хемминга (HammingDistance[u,v] - дает количество элементов, значения которых не совпадают в u и v.).

In[16]:=
```
lowstr2 = toLowCase[myString1]
```

Out[16]=
```
глава 3. современные симметричные криптосистемы
по мнению к. шеннона [101], в практических шифрах необходимо
  использовать два общих принципа: рассеивание и перемешивание.

рассеивание представляет собой распространение влияния
  одного знака открытого текста на много знаков шифртекста,
  что позволяет скрыть статистические свойства открытого текста.

перемешивание предполагает использование таких шифрующих преобразований,
  которые усложняют восстановление взаимосвязи статистических свойств
  открытого и шифрованного текстов. однако шифр должен не только
  затруднять раскрытие, но и обеспечивать легкость зашифрования
  и расшифрования при известном пользователю секретном ключе.

распространенным способом достижения эффектов рассеивания
  и перемешивания является использование составного шифра, т.е.
  такого шифра, который может быть реализован в виде некоторой
  последовательности простых шифров, каждый из которых вносит
  свой вклад в значительное суммарное рассеивание и перемешивание.

в составных шифрах в качестве простых шифров чаще всего используются простые
  перестановки и подстановки. при перестановке просто перемешивают символы
  открытого текста, причем конкретный вид перемешивания определяется
  секретным ключом. при подстановке каждый символ открытого текста
  заменяют другим символом из того же алфавита, а конкретный вид
  подстановки также определяется секретным ключом. следует заметить,
  что в современном блочном шифре блоки открытого текста и шифртекста
  представляют собой двоичные последовательности обычно длиной 64 бита. в
  принципе каждый блок может принимать 264 значений. поэтому подстановки
  выполняются в очень большом алфавите, содержащем до 264 ⌊ 1019 символов.
```

при многократном чередовании простых перестановок и подстановок, управляемых достаточно длинным секретным ключом, можно получить очень стойкий шифр с хорошим рассеиванием и перемешиванием. рассмотренные ниже криптоалгоритмы DES, IDEA и отечественный стандарт шифрования данных построены в полном соответствии с указанной методологией.

3.1. американский стандарт шифрования данных DES

стандарт шифрования данных DES (Data Encryption Standard) опубликован в 1977 г. национальным бюро стандартов сша. стандарт DES предназначен для защиты от несанкционированного доступа к важной, но несекретной информации в государственных и коммерческих организациях сша. алгоритм, положенный в основу стандарта, распространялся достаточно быстро, и уже в 1980 г. был одобрен национальным институтом стандартов и технологий сша (нист). с этого момента DES превращается в стандарт не только по названию (Data Encryption Standard), но и фактически. появляются программное обеспечение и специализированные микроэвм, предназначенные для шифрования и расшифрования информации в сетях передачи данных.

к настоящему времени DES является наиболее распространенным алгоритмом, используемым в системах защиты коммерческой информации. более того, реализация алгоритма DES в таких системах становится признаком хорошего тона.

основные достоинства алгоритма DES:

используется только один ключ длиной 56 бит;

зашифровав сообщение с помощью одного пакета программ, для расшифровки можно использовать любой другой пакет программ, соответствующий стандарту DES;

относительная простота алгоритма обеспечивает высокую скорость обработки;

достаточно высокая стойкость алгоритма.

первоначально метод, лежащий в основе стандарта DES, был разработан фирмой IBM для своих целей и реализован в виде системы люцифер. система люцифер основана на комбинировании методов подстановки и перестановки и состоит из чередующейся последовательности блоков перестановки и подстановки. в ней использовался ключ длиной 128 бит, управлявший состояниями блоков перестановки и подстановки. система люцифер оказалась весьма сложной для практической реализации из-за относительно малой скорости шифрования (2190

байт/с – программная реализация, 96970 байт/с – аппаратная реализация).

алгоритм DES также использует комбинацию подстановок и перестановок. DES осуществляет шифрование 64-битовых блоков данных с помощью 64-битового ключа, в котором значащими являются 56 бит (остальные 8 бит – проверочные биты для контроля на четность). дешифрование в DES является операцией, обратной шифрованию, и выполняется путем повторения операций шифрования в обратной последовательности. обобщенная схема процесса шифрования в алгоритме DES показана на рис.3.1. процесс шифрования заключается в начальной перестановке битов 64-битового блока, шестнадцати циклах шифрования и, наконец, в конечной перестановке битов.

In[17]:=
```
HammingDistance[lowstr1, lowstr2]
```

Out[17]=
```
0
```

12. Определить число символов "." в модифицированной строке: ⌣ StringCount[].

In[18]:=
```
StringCount[lowstr2, "."]
```

Out[18]=
```
40
```

13. Определить позиции размещения "." в модифицированной строке: ⌣ StringPosition[], получить номер позиции в виде числа.

In[19]:=
```
StringPosition[lowstr2, "."]
```

Out[19]=
```
{{8, 8}, {60, 60}, {170, 170}, {347, 347}, {523, 523}, {674, 674}, {793, 793},
 {795, 795}, {981, 981}, {1087, 1087}, {1217, 1217}, {1376, 1376}, {1532, 1532},
 {1585, 1585}, {1678, 1678}, {1868, 1868}, {2014, 2014}, {2018, 2018},
 {2020, 2020}, {2145, 2145}, {2179, 2179}, {2335, 2335}, {2427, 2427},
 {2499, 2499}, {2610, 2610}, {2759, 2759}, {2888, 2888}, {2978, 2978},
 {3328, 3328}, {3461, 3461}, {3619, 3619}, {3719, 3719}, {3911, 3911}, {3981, 3981},
 {4162, 4162}, {4304, 4304}, {4374, 4374}, {4376, 4376}, {4378, 4378}, {4532, 4532}}
```

14. Из строки, в которой была произведена замена прописных букв (см. п.11), исключить все символы (знаки препинания, цифры и т.д.), кроме строчных букв. Полученный список преобразовать в новую строку и определить ее длину: ⌣ StringCases[].

In[20]:=
```
list = StringCases[lowstr2, smallLet]
stringOnlyLetters = StringJoin[list]
```

Out[20]=

{г, л, а, в, а, с, о, в, р, е, м, е, н, н, ы, е, с, и, м, м, е, т, р,
и, ч, н, ы, е, к, р, и, п, т, о, ··· 3651 ···, и, н, а, к, о, н, е, ц, в,
к, о, н, е, ч, н, о, й, п, е, р, е, с, т, а, н, о, в, к, е, б, и, т, о, в}

large output | show less | show more | show all | set size limit ...

Out[21]=

главасовременныесимметричныекриптосистемыпомнениюкшеннонавпрактическихшифра˙.
хнеобходимоиспользоватьдваобщихпринципарассеиваниеиперемешиваниерассеивани˙.
епредставляетсобойраспространениевлияниядногознакаоткрытоготекстанамногозна˙.
ковшифртекстачтопозволяетскрытьстатистическиесвойстваоткрытоготекстаперемеш˙.
иваниепредполагаетиспользованиетакихшифрующихпреобразованийкоторыеусложня˙.
ютвосстановлениевзаимосвязистатистическихсвойствоткрытогоишифрованноготексто˙.
воднакошифрдолженнетолькозатруднятьраскрытиеноиобеспечиватьлегкостьзашифр˙.
ованияирасшифрованияприизвестномпользователюсекретномключераспространенны˙.
мспособомдостиженияэффектоврассеиванияиперемешиванияявляетсяиспользованиес˙.
оставногошифратетакогошифракоторыйможетбытьреализованввиденекоторойпослед˙.
овательностипростыхшифровкаждыйизкоторыхвноситсвойвкладвзначительноесуммар˙.
ноерассеиваниеиперемешиваниевсоставныхшифрахвкачествепростыхшифровчащевсе˙.
гоиспользуютсяпростыеперестановкиподстановкиприперестановкепростоперемешив˙.
аютсимволыоткрытоготекстапричемконкретныйвидперемешиванияопределяетсясекр˙.
етнымключомприподстановкекаждыйсимволоткрытоготекстазаменяютдругимсимволо˙.
мизтогожеалфавитаконкретныйвидподстановкитакжеопределяетсясекретнымключо˙.
мследуетзаметитьчтовсовременномблочномшифреблокиоткрытоготекстаишифртекст˙.
апредставляютсобойдвоичныепоследовательностиобычнодлинойбитавпринципекажд˙.
ыйблокможетприниматьзначенийпоэтомуподстановкивыполняютсявоченьбольшомал˙.
фавитесодержащемдосимволовпримногократномчередованииипростыхперестановокип˙.
одстановокуправляемыхдостаточнодлиннымсекретнымключомможнополучитьоченьст˙.
ойкийшифрсхорошимрассеиваниемиперемешиваниемрассмотренныенижекриптоалго˙.
ритмыиотечественныйстандартшифрованияданныхпостроенывполномсоответствиису˙.
казаннойметодологиейамериканскийстандартшифрованияданныхстандартшифрован˙.
ияданныхопубликованвгнациональнымбюростандартовсшастандартпредназначендля˙.
защитыотнесанкционированногодоступакважнойнонесекретнойинформациивгосударс˙.
твенныхикоммерческихорганизацияхсшаалгоритмположенныйвосновустандартараспр˙.
остранялсядостаточнобыстроиужевгбылодобреннациональныминститутомстандартов˙.
итехнологийсшанистсэтогомоментапревращаетсявстандартнетолькопоназваниюноиф˙.
актическипоявляютсяпрограммноеобеспечениеиспециализированныемикроэвмпредна˙.
значенныедляшифрованияирасшифрованияинформациивсетяхпередачиданныхкнасто˙.
ящемувремениявляетсянаиболеераспространеннымалгоритмомиспользуемымвсистем˙.
ахзащитыкоммерческойинформациииболеетогореализацияалгоритмавтакихсистемахст˙.
ановитсяпризнакомхорошеготонаосновныедостоинстваалгоритмаиспользуетсятолько˙.
одинключдлинойбитзашифровавсообщениеспомощьюодногопакетапрограммдлярасш˙.
ифровкиможноиспользоватьлюбойдругойпакетпрограммсоответствующийстандартуо˙.

тносительнаяпростотаалгоритмаобеспечиваетвысокуюскоростьобработкидостаточнов˙.
ысокаястойкостьалгоритмапервоначальнометодлежащийвосновестандартабылразраб˙.
отанфирмойдлясвоихцелейиреализованввидесистемылюциферсистемалюцифероснов˙.
ананакомбинированииметодовподстановкиперестановкиисостоитизчередующейсяпо˙.
следовательностиблоковперестановкиподстановкивнейиспользовалсяключдлинойби˙.
туправлявшийсостояниямиблоковперестановкиподстановкисистемалюцифероказала˙.
сьвесьмасложнойдляпрактическойреализациииззаотносительномалойскоростишифро˙.
ваниябайтспрограммнаяреализациябайтсаппаратнаяреализацияалгоритмтакжеиспол˙.
ьзуеткомбинациюподстановокиперестановокосуществляетшифрованиебитовыхблоков˙.
данныхспомощьюбитовогоключавкоторомзначащимиявляютсябитостальныебитпровер˙.
очныебитыдляконтроляначетностьдешифрованиевявляетсяоперациейобратнойшифро˙.
ваниюивыполняетсяпутемповторенияопераций шифрованиявобратнойпоследовательн˙.
остиобобщеннаясхемапроцессашифрованиявалгоритмепоказананариспроцессшифров˙.
аниязаключаетсявначальнойперестановкебитовбитовогоблокашестнадцатициклахши˙.
фрованияинаконецвконечнойперестановкебитов

15. Сформировать из списка строчных букв русского алфавита матрицу mAlfru, состоящую из 4-х строк и 8-и столбцов. Применить функцию Partition[]. Полученный результат вывести на экран в матричной форме mAlfru//MatrixForm.

In[22]:=
```
mAlfru = Partition[smallLet , 8]
mAlfru // MatrixForm
```

Out[22]=
```
{{а, б, в, г, д, е, ж, з}, {и, й, к, л, м, н, о, п},
 {р, с, т, у, ф, х, ц, ч}, {ш, щ, ъ, ы, ь, э, ю, я}}
```

Out[23]//MatrixForm=

а	б	в	г	д	е	ж	з
и	й	к	л	м	н	о	п
р	с	т	у	ф	х	ц	ч
ш	щ	ъ	ы	ь	э	ю	я

16. Проверить размерность полученного списка:→ Dimensions[].

In[24]:=
```
Dimensions[mAlfru]
```

Out[24]=
```
{4, 8}
```

17. Сформировать массив, который имеет 4на8 элементов с нулевыми начальными индексами (index origins):→Array[f,dims, origins], f – имя массива, в том числе и его элементов.

In[25]:=
```
Massiv = Array[mass , {4, 8}, 0]
```

Out[25]=
```
{{mass[0, 0], mass[0, 1], mass[0, 2], mass[0, 3], mass[0, 4], mass[0, 5], mass[0, 6], mass[0, 7]},
 {mass[1, 0], mass[1, 1], mass[1, 2], mass[1, 3], mass[1, 4], mass[1, 5], mass[1, 6], mass[1, 7]},
 {mass[2, 0], mass[2, 1], mass[2, 2], mass[2, 3], mass[2, 4], mass[2, 5], mass[2, 6], mass[2, 7]},
 {mass[3, 0], mass[3, 1], mass[3, 2], mass[3, 3], mass[3, 4], mass[3, 5], mass[3, 6], mass[3, 7]}}
```

18. Провести инициализацию элементов массива, присвоив каждому элементу массива соответствующее значение элемента матрицы, например a[0,0]= mAlfru [[1,1]] и т.д. Применить оператор Do[] для организации цикла.

In[26]:=
```
Do[mass[i, j] = mAlfru[[i + 1, j + 1]], {i, 0, 3}, {j, 0, 7}]
```

19. Вывести на экран элементы массива, представленные в табличной форме Array[,,]//Table-Form.

In[27]:=
```
Massiv // TableForm
```

Out[27]//TableForm=

а	б	в	г	д	е	ж	з
и	й	к	л	м	н	о	п
р	с	т	у	ф	х	ц	ч
ш	щ	ъ	ы	ь	э	ю	я

20. Установить ГСЧ (генератор случайных чисел) в начальное состояние с параметром - буквой русского алфавита, соответствующей номеру по списку в группе.

In[28]:=
```
SeedRandom["a"]
```

Out[28]=
```
RandomGeneratorState[ Method : ExtendedCA
                      State hash : -5 268 041 500 427 794 895 ]
```

21. Провести операции случайной перестановки элементов для первой строки , а затем второго столбца матрицы п.15:‿>RandomSample[]. Применить установку All. Проверить наличие изменений содержимого матрицы.

In[29]:=
```
mAlfru[[1, All]] = RandomSample[mAlfru[[1, All]], 8]
```

Out[29]=
```
{в, е, г, б, а, д, ж, з}
```

In[30]:=
```
mAlfru[[All, 2]] = RandomSample[mAlfru[[All, 2]], 4]
```

Out[30]=
```
{й, с, е, щ}
```

In[31]:=
```
mAlfru // MatrixForm
```

Out[31]//MatrixForm=

$$\begin{pmatrix} в & й & г & б & а & д & ж & з \\ и & с & к & л & м & н & о & п \\ р & е & т & у & ф & х & ц & ч \\ ш & щ & ъ & ы & ь & э & ю & я \end{pmatrix}$$

22. Провести повторную инициализацию массива элементами случайной перестановки матрицы.

In[32]:=
```
Do[mass[i, j] = mAlfru[[i + 1, j + 1]], {i, 0, 3}, {j, 0, 7}]
Massiv // TableForm
```

Out[33]//TableForm=

в	й	г	б	а	д	ж	з
и	с	к	л	м	н	о	п
р	е	т	у	ф	х	ц	ч
ш	щ	ъ	ы	ь	э	ю	я

23. Вывести список, состоящий из элементов строки массива с номером Nmod3 и столбца массива с номером Nmod7.

In[40]:=
```
list = Join[Massiv[[1]], Massiv[[All, 1]]
```

Out[40]= {в, й, г, б, а, д, ж, з, в, и, р, ш}

24. Вывести отображение массива в виде таблицы с границами (сетки): ⌣ Grid[Array[-ma,{4,8},0],Frame→All].

In[41]:=
```
Grid[Massiv , Frame → All]
```

Out[41]=

в	й	г	б	а	д	ж	з
и	с	к	л	м	н	о	п
р	е	т	у	ф	х	ц	ч
ш	щ	ъ	ы	ь	э	ю	я

25. Программным способом создать таблицу (матрицу) соответствия между русским алфавитом и множеством целых Z32 = {0,2,3,...,31}.

In[48]:=
```
Z32 = Range[0, 31]
```

Out[48]=
```
{0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15,
 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31}
```

In[57]:=
```
Grid[{smallLet , Z32}, Frame → All]
```

Out[57]=

а	б	в	г	д	е	ж	з	и	й	к	л	м	н	о	п	р	с	т	у	ф
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20

26. Программным способом найти элемент массива, расположенный в столбце на одну позицию ниже буквы, соответствующей номеру по списку в алфавите: ⌣ Array[],Position[],Part[],-Mod[]. В том случае, если буква, соответствующая номеру находится в нижней строке – перейти на верхнюю.

In[63]:=
```
Massiv // TableForm
```

Out[63]//TableForm=

в	й	г	б	а	д	ж	з
и	с	к	л	м	н	о	п
р	е	т	у	ф	х	ц	ч
ш	щ	ъ	ы	ь	э	ю	я

In[67]:=
```
pos = Position[Massiv, "а"]
```

Out[67]= {{1, 5}}

In[73]:=
```
x = Part[Part[pos, 1], 1] + 1
y = Part[Part[pos, 1], 2]
```

Out[73]= 2

Out[74]= 5

In[75]:=
```
Massiv[[x, y]]
```

Out[75]= м

27. Программным способом найти элемент массива, расположенный в строке справа от буквы, соответствующей номеру по списку в алфавите: ◡Array[],Position[],Part[],Mod[]. В том случае, если буква, соответствующая номеру находится в крайнем справа столбце – перейти на крайний слева.

In[76]:=
```
Massiv // TableForm
```

Out[76]//TableForm=

в	й	г	б	а	д	ж	з
и	с	к	л	м	н	о	п
р	е	т	у	ф	х	ц	ч
ш	щ	ъ	ы	ь	э	ю	я

In[79]:=
```
x = Part[Part[pos, 1], 1]
y = Part[Part[pos, 1], 2] + 1
```

Out[79]= 1

Out[80]= 6

In[82]:=
```
Massiv[[x, y]]
```

Out[82]= д