**МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ**

НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ «МЭИ»

А.А. Рытов, И.А. Яшин

ШИФРЫ ПЕРЕСТАНОВОК

Лабораторная работа №2

Учебно-методическое пособие по курсу

"Защита информации"

для студентов, обучающихся по направлению

"Информатика и вычислительная техника"

МОСКВА

Издательство МЭИ

2014

Лабораторная работа №

ШИФРЫ ПЕРЕСТАНОВКИ

Цель

Целью данной лабораторной работы является ознакомление и приобретения навыков реализации процессов шифрования и дешифрования текстов.

Подготовка к работе

Ознакомиться с теоретической справкой по шифрам перестановки и их реализации на системе Wolfram Mathematica.

Рабочее задание

1. Подготовить открытый текст для шифрования : импортировать файл соответствующий номеру N по списку в группе и содержащий открытый тест (папка Plaintext; . distributives\ импорт открытого текста.nb ), привести размер текста к величине кратной 10, удалить пробелы.
2. Определить следующие характеристики текста: энтропию позначной модели, энтропию биграммной модели, частоты чередования гласных и согласных букв (см. distributives\ анализ текста на чередование букв.nb).
3. Сформировать матрицу, содержащую 10 столбцов.
4. Провести операцию шифрования методом простой маршрутной перестановки: заполнить таблицу открытым текстом последовательно по строкам, а затем считать текст последовательно по столбцам. Определить характеристики зашифрованного текста согласно п.2.
5. Расшифровать текст соответствующий номеру N (папка Crypttext; . distributives\ импорт зашифрованного текста.nb), который зашифрован методом перестановки столбцов (шифр вертикальной перестановки) с ключом, приведенном в таблице:

|  |  |  |  |
| --- | --- | --- | --- |
| N | Ключ | N | Ключ |
| 1 | барокамера | 7 | кавалерист |
| 2 | ватерлиния | 8 | легкоатлет |
| 3 | галантерея | 9 | магнитофон |
| 4 | двухтомник | 10 | нормировка |
| 5 | жилплощадь | 11 | радиолампа |
| 6 | заповедник | 12 | стекловата |

Определить характеристики зашифрованного текста согласно п.2.

1. Зашифровать открытый текст (п. 1) с применением решетки Кардано: сформировать решетку состоящую из 10 столбцов, число строк (должно быть четным) определяется размером шифруемого текста, выбор варианта решетки определяется N (см. distributives\ формирование решетки Кардано.nb); полностью заполнить матрицу символами открытого текста (1 - позиция размещения символа), выполняя поворот решетки вокруг горизонтальной и вертикальной осей симметрии. Провести считывание последовательно по столбцам. Определить характеристики зашифрованного текста согласно п.2.
2. Провести процедуру шифрования открытого текста (п. 1) с применением шифра перестановки степени *n* , где *n* – это длина текста. Для чего сформировать случайную неповторяющуюся последовательность целых чисел из интервала [1,n] c помощью функции RandomSample[], ключом является начальное состояние генератора случайных чисел, определяемое N. Провести перестановку символов в списке и получить зашифрованный текст. Определить характеристики зашифрованного текста согласно п.2. Провести расшифрование зашифрованного текста.
3. Построить таблицу результатов экспериментов :

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | Энтропия  позначная | Энтропия  биграмм | Частота  ГГ | Частота  ГС | Частота  СГ | Частота  СС |
| Открытый  текст |  |  |  |  |  |  |
| Простая маршрутная  перестановка |  |  |  |  |  |  |
| Шифртекст перестановки по столбцам (п.5) |  |  |  |  |  |  |
| Решетка  Кардано |  |  |  |  |  |  |
| Перестановка степени *n* |  |  |  |  |  |  |

Шифры перестановки – взаимно-однозначные преобразования позиций символов исходного текста в позиции символов шифртекста.

**Оценка возможных вариантов перестановок**

**Перестановки без повторений**

Перестановками из *n* элементов (или *n-*перестановками) называют всевозможные *n-*расстановки, каждая из которых содержит все эти элементы по одному разу и которые отличаются друг от друга лишь порядком элементов.

Число *n-*перестановок обозначают через *Pn*[1]:

*Pn=n*!=1∙2∙3∙…∙n

при этом полагают, что 0!=1 и 1!=1.

В системе *Wolfram Mathematica* факториал вычисляется с помощью функции ***Factorial*(!)**. Пример результатов расчета функции *x*! для x=0..10 представлен в листинге 1.

Листинг 1. Расчет факториала

**Table[***x*!,**{***x*,0,10}]

{1, 1, 2, 6, 24, 120, 720, 5040, 40 320, 362 880, 3 628 800}

При больших *n* для приближенного вычисления *n*! можно воспользоваться формулой Стирлинга:

где *e*=2.718281828…

Пример реализации всех возможных перестановок для слова *лань* приведен в листинге 2. Поскольку все буквы в этом слове различные, реализованы все 4!=24 перестановки. Справка по функции ***Permutations*** приведена в Приложении 1.

Листинг 2. Перестановки слова *лань*

**Permutations[Characters[***“лань”***]]**

{{л,а,н,ь},{л,а,ь,н},{л,н,а,ь},{л,н,ь,а},{л,ь,а,н},{л,ь,н,а},{а,л,н,ь},  
{а,л,ь,н},{а,н,л,ь},{а,н,ь,л},{а,ь,л,н},{а,ь,н,л},{н,л,а,ь},{н,л,ь,а},  
{н,а,л,ь},{н,а,ь,л},{н,ь,л,а},{н,ь,а,л},{ь,л,а,н},{ь,л,н,а},{ь,а,л,н},  
{ь,а,н,л},{ь,н,л,а},{ь,н,а,л}}

**Перестановки с повторениями**

В том случае, если некоторые переставляемые элементы (символы текста) одинаковы, то число возможных различаемых перестановок уменьшается, так как некоторые перестановки совпадают друг с другом.

Например, если в слове из четырех букв есть две одинаковые буквы, число возможных перестановок сокращается с 24 до 12 (листинг 3).

Листинг 3. Перестановки слова *мусс*

**Permutations[Characters[***“мусс”***]]**

{{м,у,с,с},{м,с,у,с},{м,с,с,у},{у,м,с,с},{у,с,м,с},{у,с,с,м},{с,м,у,с},  
{с,м,с,у},{с,у,м,с},{с,у,с,м},{с,с,м,у},{с,с,у,м}}

Для четырехбуквенного слова, в котором присутствуют две одинаковые буквы число перестановок сокращается до 8 (листинг 4.)

Листинг 4. Перестановки слова *мама*

**Permutations[Characters[***“мама”***]]**

{{м,а,м,а},{м,а,а,м},{м,м,а,а},{а,м,м,а},{а,м,а,м},{а,а,м,м}}

В общем случае число перестановок с повторениями определяется следующим образом:

Таким образом легко определить число возможных перестановок для некоторого произвольного текста.

Строка

str=”автоматизированныесистемыобработкиинформациииуправления” содержит 55 букв и если бы повторов не было, число возможных вариантов перестановок было бы равно:

55!= 12696403353658275925965100847566516959580321051449436762275840000000000000 ≈ 1,26964∙1073

Однако, подсчет частот появления букв из которых состоит эта строка дает следующие результаты: (листинг 5)

Листинг 5 – подсчет частот появления букв в строке

s=**Tally[Characters[**str**]]**

{{а,6},{в,3},{т,4},{о,5},{м,3},{и,9},{з,1},{р,4},{н,4},{ы,2},{е,3},{с,2},{б,2},{к,1},{ф,1},{ц,1},{у,1},{п,1},{л,1},{я,1}}

То есть, буква “a” встречается 6 раз, буква “в” – 3 раза и так далее, поэтому число возможных перестановок будет равно: (листинг 6)

Листинг 6 – Расчет числа перестановок в строке

16952215808839053268245850144092368988466728960000000000

Что приблизительно равно 1,69522∙1055. Таким количеством из-за повторяющихся букв число возможных вариантов уменьшилось в 1018 раз.

**Генерация перестановок.**

Существует множество различных алгоритмов генерации перестановок. Основные из них приводятся в [2].

Одним из них является алгоритм Нарайана или *L*-алгоритм в обозначениях Д. Кнута [2]. Разработан Пандита Нарайана в 14 веке в Индии.

Для последовательности из *n* элементов *a1a2….an*, в начальном состоянии отсортированном так, что *a1≤a2≤…≤an*, этот алгоритм генерирует все перестановки {*a1,a2,…,an*} в их лексикографическом порядке.

Например, перестановки последовательности {1,2,2,3} имеют следующий лексикографический порядок:

{{1,2,2,3},{1,2,3,2},{1,3,2,2},{2,1,2,3},{2,1,3,2},{2,2,1,3},{2,2,3,1},  
{2,3,1,2},{2,3,2,1},{3,1,2,2},{3,2,1,2},{3,2,2,1}}

Алгоритм состоит из следующих шагов [2]:

1. Вызвать перестановку *a1a2…an*
2. Установить *j=n-1.* Если *aj≥aj+1*, уменьшать *j* на 1, пока не будет выполнено условие *aj<aj+1.* Завершить алгоритм, если *j=*0.
3. Установить *l*=*n*. Если *aj≥al*, уменьшать *l* на 1, пока не будет выполнено условие *aj<al*. Затем поменять местами *aj* и *al.*
4. Развернуть последовательность *aj+1*,…,*an.* Вернуться к пункту 1.

Программная реализация алгоритма Нарайна в системе *Wolfram* *Mathematica* приведена в приложении 2.

Кроме сторонних алгоритмов перестановка может быть сгенерирована уже упомянутой выше встроенной функцией ***Permutations*[]** (Приложение 1), а также встроенной функцией ***RandomSample*[]** (Листинг 7)

Листинг 7. Генерация перестановки с помощью RandomSamle для списка из первых ста натуральных чисел

**RandomSample[Range[**100**]]**

{58,83,23,100,79,59,90,13,60,93,37,52,17,43,4,97,53,39,86,69,66,40,63,  
26,22,50,10,8,2,11,96,18,29,92,47,85,95,55,14,99,38,67,1,25,9,73,3,71,56,49,36,81,28,74,20,51,84,54,33,19,76,89,30,24,42,48,34,80,57,5,16,7,82,41,15,31,87,88,6,98,46,21,62,70,45,68,94,35,72,61,78,44,65,64,12,32,77,91,  
27,75}

**Шифры перестановки**

**Подстановка степени *n***

Этот метод используется для шифрования сообщения, состоящего из *n* символов. На первом этапе формируется таблица вида:

В первой строке расположены индексы символов открытого текста, а во второй – ключ, содержащий позиции шифр текста, на которые попадает соответствующая буква открытого текста. Ключ формируется как случайная перестановка чисел от 1 до *n*. Такая таблица называется подстановкой степени *n* [3].

Пример реализации данного шифра в системе *Mathematica* приведен в листинге 8.

Листинг 8. Реализация шифра подстановки степени *n*

(\*Первый этап – формирование списка символов открытого текста\*)

***openText*=**"шифрыперестановки"

шифры перестановки

***listText***=***Characters*[*openText*]**

{ш,и,ф,р,ы, ,п,е,р,е,с,т,а,н,о,в,к,и}

(\*Второй этап – генерация псевдослучайного ключа. Для повторяемости результатов применена функция *SeedRandom*[]\*)

***SeedRandom*[**100**]**

***keyn*=*RandomSample*[*Range*[*Length*[*listText*]]]**

{1,14,3,17,13,9,8,15,10,4,7,6,11,16,12,18,2,5}

***cryptText***={};

***Do*[*AppendTo*[*cryptText***,***listText***[[***keyn*[[***i***]]]]]**,{*i*,**Length[***listText***]**}**]**

***StringJoin*[*cryptText*]**

шнфкареоерп свтииы

Информация о позициях открытого текста содержится в ключе шифрования *keyn*, поэтому, зная ключ и выполняя обратные операции по отношению к процессу шифрования, можно восстановить исходный текст (Листинг 9)

Листинг 9. Реализация расшифровки текста, зашифрованного перестановками степени *n*

***str***={};

***Do*[*AppendTo*[*str***,***cryptText*[[*Flatten*[*Position*[*keyn***,*i***]]]]]**,{i**,*Length*[*listText*]**}**]**

***StringJoin*[*str*]**

шифры перестановки

**Шифры маршрутной перестановки**

Для компоновки исходной информации используется та или иная геометрическая фигура, которая является частью ключа. Также составляющими ключа являются маршрут записи информации в выбранную фигуру и маршрут считывания. Простейшим вариантом маршрутной перестановки является табличная перестановка в котором текст записывается в прямоугольную таблицу по строками слева-направо, а считывается по столбцам сверху-вниз.

Реализация табличной перестановки для троки с текстом «автоматизированныесистемыобработкиинформации» представлен в Листинге 10 и 11. Данный текст содержит 44 буквы и может быть размещен в таблице 4х11.

Предварительная подготовка состоит в формировании списка символов текста с помощью функции ***Characters*[]**.

Формирование таблицы происходит с помощью функции ***Partition*[]**, которая разбивает список на *m* (в данном случае 4) непересекающихся строк.

Функция ***Grid*** с опцией **Frame→*All*** служит для визуализации структуры матрицы.

Листинг 10. Формирование матрицы для маршрутной перестановки

***openText***="автоматизированныесистемыобработкиинформации";

***listText***=***Characters*[*openText*]**

{а,в,т,о,м,а,т,и,з,и,р,о,в,а,н,н,ы,е,с,и,с,т,е,м,ы,о,б,р,а,б,о,т,к,и,и,н,ф,о,р,м,а,ц,и,и}

***Grid*[*matrix*=*Partition*[*listText***,11**]**,***Frame*→*All*]**

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| а | в | т | о | м | а | т | и | з | и | р |
| о | в | а | н | н | ы | е | с | и | с | т |
| е | м | ы | о | б | р | а | б | о | т | к |
| и | и | н | ф | о | р | м | а | ц | и | и |

Считывание информации по столбцам также можно осуществить несколькими способами, в том числе с помощью операции транспонирования матрицы с дальнейшим объединением текста (листинг 11)

Листинг 11. Считывание матрицы по столбцам

***Grid*[*mt*=*Transpose*[*matrix*], *Frame*→*All*]**

|  |  |  |  |
| --- | --- | --- | --- |
| а | о | е | и |
| в | в | м | и |
| т | а | ы | н |
| о | н | о | ф |
| м | н | б | о |
| а | ы | р | р |
| т | е | а | м |
| и | с | б | а |
| з | и | о | ц |
| и | с | т | и |
| р | т | к | и |

***cryptText*=*StringJoin*[*mt*]**

аоеиввмитаынонофмнбоаырртеамисбазиоцистиртки

Для того, чтобы расшифровать текст «аоеиввмитаынонофмнбоаырртеамисбазиоцистиртки» его необходимо записать в таблицу 4х11 по столбцам последовательно сверху вниз, а затем считать по строкам. Эквивалентные операции процесса расшифрования, приведены в Листинге 12.

Листинг 12. Расшифровка маршрутной перестановки

***cryptText***="аоеиввмитаынонофмнбоаырртеамисбазиоцистиртки";

**cm=*Partition*[*Characters*[*cryptText*]**,4**]**

{{а,о,е,и},{в,в,м,и},{т,а,ы,н},{о,н,о,ф},{м,н,б,о},{а,ы,р,р},{т,е,а,м},  
{и,с,б,а},{з,и,о,ц},{и,с,т,и},{р,т,к,и}}

***StringJoin*[*Transpose*[*cm*]]**

автоматизированныесистемыобработкиинформации

**Шифры столбцовой (вертикальной) перестановки**

Также как и в шифрах табличной перестановки, здесь применяется таблица, в которую вписывается текст последовательно по строкам и считывается по столбцам. Однако, между этими процессами, проводится преобразование информации, которое определяется ключевым словом, фразой и ли набором числе длиной в строку таблицы. Ключевое слово, в нашем примере «радиограмма», записывается в верхней строке таблицы:

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **р** | **а** | **д** | **и** | **о** | **г** | **р** | **а** | **м** | **м** | **а** |
| а | в | т | о | м | а | т | и | з | и | р |
| о | в | а | н | н | ы | е | с | и | с | т |
| е | м | ы | о | б | р | а | б | о | т | к |
| и | и | н | ф | о | р | м | а | ц | и | и |

Присутствующая в ключевом слове буква «а» (расположенная во втором столбце) получает номер 1. Если какая-то буква входит несколько раз, то ее вхождения нумеруются последовательно слева направо. Поскольку буквы «б» и «в» отсутствуют, то буква «г», расположенная в шестом столбце) получает номер 4 и так далее в алфавитном порядке. В результате формируется таблица перестановок столбцов матрицы (листинг 13)

Листинг 13. Формирование таблицы перестановок по ключевому слову

***slovo***="радиограмма";

***ord***=***Ordering*[*Characters*[*slovo*]]**

{2,8,11,6,3,4,9,10,5,1,7}

***Grid*[*key***={***ord***,***Range*[**11**]**},***Frame*→*All*]**

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 2 | 8 | 11 | 6 | 3 | 4 | 9 | 10 | 5 | 1 | 7 |
| 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 |

В соответствии с полученным ключом (*key*) на первое место перемещается второй столбец исходной таблицы (естественно без ключевого слова), восьмой столбец становится вторым и т.д., после чего производится считывание по столбцам.

Рассмотрим пример, в котором на базе исходной таблицы, формируется таблица шифртекста, определяемая ключом перестановки *key* (листинг 14).

Сначала таблица шифртекста инициализируется нулевыми значениями (листинг 15)

Листинг 15. Первоначальная инициализациия таблицы

***openText***="автоматизированныесистемыобработкиинформации";

***openTable***=***Partition*[*Characters*[*openText*]**,11**]**;

***Grid*[*openTable***, ***Frame*→*All*]**

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| а | в | т | о | м | а | т | и | з | и | р |
| о | в | а | н | н | ы | е | с | и | с | т |
| е | м | ы | о | б | р | а | б | о | т | к |
| и | и | н | ф | о | р | м | а | ц | и | и |

***cryptTable*=*Table*[0**,{**4**},{**11**}**];**

***Grid*[*cryptTable***, ***Frame*→*All*]**

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

Далее таблица *cryptTable* заполняется столбцами таблицы *openTable* в соответствии с ключом *key* (листинг 16)

Листинг 16. Формирование таблицы шифртекста

***Do*[*cryptTable*[[*All***,***key*[[**2,*i***]]]]=*openTable*[[*All***,***key*[[**1,*i***]]]]**,{*i*,11}]

***Row*[**{***Grid*[*openTable*,*Frame*→*All*], *Grid*[*cryptTable*,*Frame*→*All*]**}**," "]**

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| а | в | т | о | м | а | т | и | з | и | р |  | в | и | р | а | т | о | з | и | м | а | т |
| о | в | а | н | н | ы | е | с | и | с | т |  | в | с | т | ы | а | н | и | с | н | о | е |
| е | м | ы | о | б | р | а | б | о | т | к |  | м | б | к | р | ы | о | о | т | б | е | а |
| и | и | н | ф | о | р | м | а | ц | и | и |  | и | а | и | р | н | ф | ц | и | о | и | м |

Как видно из последних таблиц, в результате перестановки столбцов, второй столбец исходной таблицы *openTable* оказался в таблице шифртекста *cryptText* первым, восьмой – вторым и т.д.

При считывании их таблицы шифртекста по столбцам получаем строку шифртекста (листинг 17)

Листинг 17. Считывание шифртекста по столбцам

***cryptText***={};

***Do*[*AppendTo*[*cryptText*,*cryptTable*[[*All***,i**]]]**,{i,**11**}**]**

***cryptText*=*StringJoin*[*cryptText*]**

ввмиисбарткиаырртаынонофзиоцистимнбоаоеитеам

Операция расшифрования – это обратная операция, поэтому необходимо начинать с заполнения таблицы по столбцам и учитывать, что строки в ключе меняются местами. (Листинг 18)

Листинг 18. Формирование матрицы шифртекста

**cryptTable=Partition[Characters[cryptText]**,4**]**

{{в,в,м,и},{и,с,б,а},{р,т,к,и},{а,ы,р,р},{т,а,ы,н},{о,н,о,ф},{з,и,о,ц},  
{и,с,т,и},{м,н,б,о},{а,о,е,и},{т,е,а,м}}

**Grid[cryptTable=Transpose[cryptTable]**, **Frame→All]**

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| в | и | р | а | т | о | з | и | м | а | т |
| в | с | т | ы | а | н | и | с | н | о | е |
| м | б | к | р | ы | о | о | т | б | е | а |
| и | а | и | р | н | ф | ц | и | о | и | м |

**decryptTable=Table[**0,{4},{11}**];**

**Do[decryptTable[[All**,**key[[**1,*i***]]]]=cryptTable[[All,key[[**2,*i***]]]]**,{i,11}]

**Row[{Grid[cryptTable,Frame→All]**,**Grid[decryptTable,Frame→All]}," "]**

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| в | и | р | а | т | о | з | и | м | а | т |  | а | в | т | о | м | а | т | и | з | и | р |
| в | с | т | ы | а | н | и | с | н | о | е |  | о | в | а | н | н | ы | е | с | и | с | т |
| м | б | к | р | ы | о | о | т | б | е | а |  | е | м | ы | о | б | р | а | б | о | т | к |
| и | а | и | р | н | ф | ц | и | о | и | м |  | и | и | н | ф | о | р | м | а | ц | и | и |

В результате считывания таблицы decryptTable по строкам (см. Листинг 17) получаем исходный текст "автоматизированныесистемыобработкиинформации".

**Решетка Кардано**

Шифр решетка Кардано использует трафарет с четным (2*m*) числом строк и четным (2*k*) числом столбцов, в котором вырезано *m∙k* клеток так, что при наложении его на чистый лист бумаги того же размера четырьмя возможными способами его вырезы покрывают всю площадь листа.

Буквы шифруемого текста последовательно вписывают в вырезы трафарета по строкам, в каждой строке слева направо, Затем трафарет поворачивают вдоль вертикальной оси симметрии на 180º, а затем вдоль горизонтальной оси симметрии также на 180º, и вновь вдоль вертикальной оси симметрии на 180º (рисунок 1)

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |

Рисунок 1. Решетка Кардано 4х4

Например, для текста

"профильподготовкиавтоматизированныесистемыобработкиинформации",

длина которого составляет 72 символа, можно построить решетку Кардано размером 6х12.

В листинге 18 приведена программная реализация функции *cardano*, реализующей начальную решетку – матрицу *m*0. Входными параметрами функции являются: число строк, число столбцов и начальное состояние генератора случайных чисел.

Листинг 18. Генерация начальной решетки Кардано

**cardano[***k0*\_,*n0*\_,*srand0*\_**]:=Module[**{n=*n0*,k=*k0*,srand=*srand0*},

r=1; **SeedRandom[**srand**]**; **listZ**={};

**m0**=**Table[**0,{l,k},{m,n}**]**;

**While[**r!=n\*k/4+1,**i**=**RandomInteger[**{1,k}**]**;**j**=**RandomInteger[**{1,n}**]**;

**If**[!MemberQ**[listZ**,{i,j}**]**,**listZ**=**Append[**listZ,{i,j}**]**;

**listZ**=**Append[**listZ,{i,n-j+1}**]**;

**listZ**=**Append[**listZ,{k-i+1,j}**]**;

**listZ**=**Append[**listZ,{k-i+1,n-j+1}**]**;

r++;**m0[[**i,j**]]**=1**]]**; **m0]**

В результате выполнения данной функции с параметрами (6,12,1) генерируется начальная решетка Кардано, единичные позиции которой предназначены для записи символов из шифруемой строки (рис. 2)

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 1 | 1 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 1 | 1 | 0 | 0 |
| 0 | 1 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 1 |
| 0 | 0 | 0 | 1 | 0 | 0 | 1 | 0 | 0 | 1 | 0 | 0 |
| 0 | 1 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 1 |
| 0 | 0 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

Рисунок 2 – Начальная решетка Кардано

В листинге 19 приведены функции поворота трафарета вокруг вертикальной оси симметрии (относительно столбцов) rotColSim и горизонтальной (относительно рядов) – rotRowSim.

Листинг 19. Повороты решетки Кардано

**rotColSim[**table0\_**]**:=**Module[**{table=table0},

**k**=**Dimensions[**table**][[**1**]]**; **n**=**Dimensions[**table**][[**2**]]**;v=1;s=n;

**While[**v<=n/2,tempCol=table**[[All**,v**]]**;table**[[All**,v**]]**=table**[[All**,s**]]**;

table**[[All**,s**]]**=tempCol;v++;s--;t=table

];table]

**rotRowSim[**table0\_**]:=Module[**{table=table0},

k=**Dimensions[**table**][[**1**]]**;n=**Dimensions[**table**][[**2**]]**;v=1;s=k;

**While**[v<=k/2,tempRow=table**[[**v,All**]]**;table**[[**v,**All]]**=table**[[**s,**All]]**;

table**[[**s,**All]]**=tempRow;v++;s--;t=table

**]**;table**]**

В листинге 20 приведена функция *cardFull* генерирующая все четыре варианта трафаретов решетки Кардано с проверкой наложения.

Листинг 20. Генерация полного набора трафаретов

**cardFull[**row0\_,col0\_,srnd0\_**]**:=

**Module[**{row=row0,col=col0,srnd=srnd0},

card0=cardano[row,col,srnd];

card1=rotColSim[card0];

card2=rotRowSim[card1];

card3=rotColSim[card2];

Row[{Grid[card0,Frame**→All**], Grid[card1,Frame**→All**], Grid[card2,Frame**→All**], Grid[card3,Frame**→All**], Grid[card0+card1+card2+card3,Frame**→All**]}]]

Полный набор трафаретов *cardFull*[6,12,1] *представлен* на рисунке 3.

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 1 | 1 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 |  | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 1 | 1 |
| 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 1 | 1 | 0 | 0 |  | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 |
| 0 | 1 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 1 |  | 1 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 1 | 0 |
| 0 | 0 | 0 | 1 | 0 | 0 | 1 | 0 | 0 | 1 | 0 | 0 |  | 0 | 0 | 1 | 0 | 0 | 1 | 0 | 0 | 1 | 0 | 0 | 0 |
| 0 | 1 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 1 |  | 1 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 1 | 0 |
| 0 | 0 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |  | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 0 | 0 |

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 0 | 0 |  | 0 | 0 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 1 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 1 | 0 |  | 0 | 1 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 1 |
| 0 | 0 | 1 | 0 | 0 | 1 | 0 | 0 | 1 | 0 | 0 | 0 |  | 0 | 0 | 0 | 1 | 0 | 0 | 1 | 0 | 0 | 1 | 0 | 0 |
| 1 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 1 | 0 |  | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 1 |
| 0 | 0 | 1 | 1 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 |  | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 1 | 1 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 1 | 1 |  | 1 | 1 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 |

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |

Поскольку в последней таблице, представляющей сумму четырех возможных позиций трафарета, все ячейки заняты 1, то шифрующий текст заполнит всю таблицу.

Операция шифрования заключается в записи в соответствующую таблицу символов из строки *openText* на позиции, содержащие 1. В листинге 21 показан результат после заполнения таблицы с помощью первого трафарета.

Листинг 21. Заполнение трафарета карты Кардано

opentText="профильподготовкиавтоматизированныесистемыобработкиинформации";

s=1;

Do[If[card0[[i,j]]==1,card0[[i,j]]=StringTake[opentText,{s}];s++],{i,1,6},{j,1,6}]

Grid[card0,Frame->All]

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| п | р | 0 | 0 | 0 | о | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | ф | 0 | 0 | 0 | и | л | 0 | 0 |
| 0 | ь | 0 | 0 | 0 | 0 | 0 | п | 0 | 0 | 0 | о |
| 0 | 0 | 0 | д | 0 | 0 | г | 0 | 0 | о | 0 | 0 |
| 0 | т | 0 | 0 | 0 | о | 0 | 0 | 0 | 0 | 0 | в |
| 0 | 0 | к | и | а | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

Последовательно заполняя остальные подготовленные трафареты получаем результат шифрования текста в табличном виде (листинг 22)

Листинг 22. Генерация полной таблицы шифрования Кардано

s=1;

Do[If[card1[[i,j]]1,card1[[i,j]]=StringTake[opentText,{s}];s++],{i,1,6},{j,1,6}]

s=1;

Do[If[card2[[i,j]]1,card2[[i,j]]=StringTake[opentText,{s}];s++],{i,1,6},{j,1,6}]

s=1;

Do[If[card3[[i,j]]1,card3[[i,j]]=StringTake[opentText,{s}];s++],{i,1,6},{j,1,6}]

Grid[card0+card1+card2+card3,Frame**→All**]

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| п | р | о | р | м | о | в | и | с | т | т | о |
| е | а | м | а | ф | ц | м | т | и | л | ы | и |
| и | ь | о | и | з | б | и | п | р | у | и | о |
| а | п | р | д | б | о | г | р | в | о | о | а |
| а | т | т | к | в | о | н | и | л | е | н | в |
| н | и | к | и | а | я | и | ы | е | с | н | Ф |

Окончательный шифр текст формируется путем считывания таблицы по столбцам (см листинги 11 и 17)

**Магические квадраты**

Магический квадрат – это квадратная таблица, заполненная *n*2 натуральными числами таким образом, что суммы по всем строкам, столбцам и обеим диагоналям равны между собой.

Минимальным нетривиальным случаем магического квадрата является квадрат 3х3 (рисунок 3)

|  |  |  |
| --- | --- | --- |
| 2 | 7 | 6 |
| 9 | 5 | 1 |
| 4 | 3 | 8 |

Рисунок 3 – магический квадрат 3х3

Вопрос генерации магических квадратов высоких порядков выходит за рамки данной работы, один из алгоритмов приведен в Приложении 3.

При шифровании с использованием магических квадратов символы открытого текста расставляются в квадрат по номерам начиная с 1, затем текст считывается по строкам (листинг 22).

Листинг 22. Шифрование с помощью магического квадрата

opentText="обработкасигнала";

square={{4,14,15,1},{9,7,6,12},{5,11,10,8},{16,2,3,13}};

Grid[square,Frame->All]

|  |  |  |  |
| --- | --- | --- | --- |
| 4 | 14 | 15 | 1 |
| 9 | 7 | 6 | 12 |
| 5 | 11 | 10 | 8 |
| 16 | 2 | 3 | 13 |

cryptTable=Table[StringTake[opentText,{square[[i,j]]}],{i,4},{j,4}]

Grid[cryptTable,Frame->All]

|  |  |  |  |
| --- | --- | --- | --- |
| а | а | л | о |
| а | т | о | г |
| б | и | с | к |
| а | б | р | н |

Расшифровка текста из подготовленной таблицы осуществляется путем подстановки буквы в соответствующую позицию строки, обусловленную магическим квадратом (листинг23).

Листинг 23 -

decryptText=Table["",{4\*4}];

Do[decryptText[[square[[i,j]]]]=cryptTable[[i,j]],{i,4},{j,4}]

StringJoin[decryptText]

Обработкасигнала

Приложение 1. Анализ характеристик текста (частота ГГ, ГС, СГ, СС)

opentext="ввмиисбарткиаырртаынонофзиоцистимнбоаоеитеам";

crypttext="автоматизированныесистемыобработкиинформации";

glas=Characters["уеыаоэяию"]

{у,е,ы,а,о,э,я,и,ю}

sogl=Characters["йцкнгшщзхфвпрлджчмтб"]

{й,ц,к,н,г,ш,щ,з,х,ф,в,п,р,л,д,ж,ч,м,т,б}

gg=StringJoin/@Tuples[{glas,glas}]

{уу,уе,уы,уа,уо,уэ,уя,уи,ую,еу,ее,еы,еа,ео,еэ,ея,еи,ею,ыу,ые,ыы,ыа,ыо,ыэ,ыя,ыи,ыю,ау,ае,аы,аа,ао,аэ,ая,аи,аю,оу,ое,оы,оа,оо,оэ,оя,ои,ою,эу,эе,эы,эа,эо,ээ,эя,эи,эю,яу,яе,яы,яа,яо,яэ,яя,яи,яю,иу,ие,иы,иа,ио,иэ,ия,ии,ию,юу,юе,юы,юа,юо,юэ,юя,юи,юю}

gs=StringJoin/@Tuples[{glas,sogl}]

{уй,уц,ук,ун,уг,уш,ущ,уз,ух,уф,ув,уп,ур,ул,уд,уж,уч,ум,ут,уб,ей,ец,ек,ен,ег,еш,ещ,ез,ех,еф,ев,еп,ер,ел,ед,еж,еч,ем,ет,еб,ый,ыц,ык,ын,ыг,ыш,ыщ,ыз,ых,ыф,ыв,ып,ыр,ыл,ыд,ыж,ыч,ым,ыт,ыб,ай,ац,ак,ан,аг,аш,ащ,аз,ах,аф,ав,ап,ар,ал,ад,аж,ач,ам,ат,аб,ой,оц,ок,он,ог,ош,ощ,оз,ох,оф,ов,оп,ор,ол,од,ож,оч,ом,от,об,эй,эц,эк,эн,эг,эш,эщ,эз,эх,эф,эв,эп,эр,эл,эд,эж,эч,эм,эт,эб,яй,яц,як,ян,яг,яш,ящ,яз,ях,яф,яв,яп,яр,ял,яд,яж,яч,ям,ят,яб,ий,иц,ик,ин,иг,иш,ищ,из,их,иф,ив,ип,ир,ил,ид,иж,ич,им,ит,иб,юй,юц,юк,юн,юг,юш,ющ,юз,юх,юф,юв,юп,юр,юл,юд,юж,юч,юм,ют,юб}

sg=StringJoin/@Tuples[{sogl,glas}]

{уй,уц,ук,ун,уг,уш,ущ,уз,ух,уф,ув,уп,ур,ул,уд,уж,уч,ум,ут,уб,ей,ец,ек,ен,ег,еш,ещ,ез,ех,еф,ев,еп,ер,ел,ед,еж,еч,ем,ет,еб,ый,ыц,ык,ын,ыг,ыш,ыщ,ыз,ых,ыф,ыв,ып,ыр,ыл,ыд,ыж,ыч,ым,ыт,ыб,ай,ац,ак,ан,аг,аш,ащ,аз,ах,аф,ав,ап,ар,ал,ад,аж,ач,ам,ат,аб,ой,оц,ок,он,ог,ош,ощ,оз,ох,оф,ов,оп,ор,ол,од,ож,оч,ом,от,об,эй,эц,эк,эн,эг,эш,эщ,эз,эх,эф,эв,эп,эр,эл,эд,эж,эч,эм,эт,эб,яй,яц,як,ян,яг,яш,ящ,яз,ях,яф,яв,яп,яр,ял,яд,яж,яч,ям,ят,яб,ий,иц,ик,ин,иг,иш,ищ,из,их,иф,ив,ип,ир,ил,ид,иж,ич,им,ит,иб,юй,юц,юк,юн,юг,юш,ющ,юз,юх,юф,юв,юп,юр,юл,юд,юж,юч,юм,ют,юб}

sg=StringJoin/@Tuples[{sogl,glas}]

{йу,йе,йы,йа,йо,йэ,йя,йи,йю,цу,це,цы,ца,цо,цэ,ця,ци,цю,ку,ке,кы,ка,ко,кэ,кя,ки,кю,ну,не,ны,на,но,нэ,ня,ни,ню,гу,ге,гы,га,го,гэ,гя,ги,гю,шу,ше,шы,ша,шо,шэ,шя,ши,шю,щу,ще,щы,ща,що,щэ,щя,щи,щю,зу,зе,зы,за,зо,зэ,зя,зи,зю,ху,хе,хы,ха,хо,хэ,хя,хи,хю,фу,фе,фы,фа,фо,фэ,фя,фи,фю,ву,ве,вы,ва,во,вэ,вя,ви,вю,пу,пе,пы,па,по,пэ,пя,пи,пю,ру,ре,ры,ра,ро,рэ,ря,ри,рю,лу,ле,лы,ла,ло,лэ,ля,ли,лю,ду,де,ды,да,до,дэ,дя,ди,дю,жу,же,жы,жа,жо,жэ,жя,жи,жю,чу,че,чы,ча,чо,чэ,чя,чи,чю,му,ме,мы,ма,мо,мэ,мя,ми,мю,ту,те,ты,та,то,тэ,тя,ти,тю,бу,бе,бы,ба,бо,бэ,бя,би,бю}

ss=StringJoin/@Tuples[{sogl,sogl}]

{йй,йц,йк,йн,йг,йш,йщ,йз,йх,йф,йв,йп,йр,йл,йд,йж,йч,йм,йт,йб,цй,цц,цк,цн,цг,цш,цщ,цз,цх,цф,цв,цп,цр,цл,цд,цж,цч,цм,цт,цб,кй,кц,кк,кн,кг,кш,кщ,кз,кх,кф,кв,кп,кр,кл,кд,кж,кч,км,кт,кб,нй,нц,нк,нн,нг,нш,нщ,нз,нх,нф,нв,нп,нр,нл,нд,нж,нч,нм,нт,нб,гй,гц,гк,гн,гг,гш,гщ,гз,гх,гф,гв,гп,гр,гл,гд,гж,гч,гм,гт,гб,шй,шц,шк,шн,шг,шш,шщ,шз,шх,шф,шв,шп,шр,шл,шд,шж,шч,шм,шт,шб,щй,щц,щк,щн,щг,щш,щщ,щз,щх,щф,щв,щп,щр,щл,щд,щж,щч,щм,щт,щб,зй,зц,зк,зн,зг,зш,зщ,зз,зх,зф,зв,зп,зр,зл,зд,зж,зч,зм,зт,зб,хй,хц,хк,хн,хг,хш,хщ,хз,хх,хф,хв,хп,хр,хл,хд,хж,хч,хм,хт,хб,фй,фц,фк,фн,фг,фш,фщ,фз,фх,фф,фв,фп,фр,фл,фд,фж,фч,фм,фт,фб,вй,вц,вк,вн,вг,вш,вщ,вз,вх,вф,вв,вп,вр,вл,вд,вж,вч,вм,вт,вб,пй,пц,пк,пн,пг,пш,пщ,пз,пх,пф,пв,пп,пр,пл,пд,пж,пч,пм,пт,пб,рй,рц,рк,рн,рг,рш,рщ,рз,рх,рф,рв,рп,рр,рл,рд,рж,рч,рм,рт,рб,лй,лц,лк,лн,лг,лш,лщ,лз,лх,лф,лв,лп,лр,лл,лд,лж,лч,лм,лт,лб,дй,дц,дк,дн,дг,дш,дщ,дз,дх,дф,дв,дп,др,дл,дд,дж,дч,дм,дт,дб,жй,жц,жк,жн,жг,жш,жщ,жз,жх,жф,жв,жп,жр,жл,жд,жж,жч,жм,жт,жб,чй,чц,чк,чн,чг,чш,чщ,чз,чх,чф,чв,чп,чр,чл,чд,чж,чч,чм,чт,чб,мй,мц,мк,мн,мг,мш,мщ,мз,мх,мф,мв,мп,мр,мл,мд,мж,мч,мм,мт,мб,тй,тц,тк,тн,тг,тш,тщ,тз,тх,тф,тв,тп,тр,тл,тд,тж,тч,тм,тт,тб,бй,бц,бк,бн,бг,бш,бщ,бз,бх,бф,бв,бп,бр,бл,бд,бж,бч,бм,бт,бб}

Length@StringPosition[opentext,gg]

5

Length@StringPosition[crypttext,gg]

4

Приложение 2. Алгоритм Нарайана

(\*Одна перестановка\*)

**NarajanaStep**[*list*\_]:=**Module[**{*j,l,temp,res*},

*res*=*list*;

*j*=**Length[***res***]**-1;

**While[**!(res[[j]]<res[[j+1]])&&j>1,j=j-1**]**;

l=**Length**[res];

**While**[res[[j]]>=res[[l]]&&l>1,l=l-1];

temp=res[[l]];

res[[l]]=res[[j]];

res[[j]]=temp;

res[[j+1;;-1]]=**Reverse**[res[[j+1;;-1]]];

res

]

**NarajanaAll**[list\_]:=**Module[**{res,k},

res={list};

k=**Length[First**@res**]**!;

**Do[AppendTo[**res,**NarajanaStep[Last**@res**]]**,{k-1}**]**;

res]

Приложение 3. Функция Permutation

Перестановки в системе Wolfram Mathematica представлены рядом функций.

Простейшей из них является функция **Permutations**[].

Базовый синтаксис:

**Permutations**[список элементов]

При таком синтаксисе функция выдает список всех возможных комбинаций такой же мощности как и мощность входного списка, например:

p=Permutations[{1,2,3}]

{{1,2,3},{1,3,2},{2,1,3},{2,3,1},{3,1,2},{3,2,1}}

Вычислив длину получившегося списка можно убедиться, что количество перестановок мощности n есть факториал n.

Length[p]

6

При указании второго параметра функции **Permutation**[] можно задать длину выходных последовательностей, так конструкция

Permutation[список, n] вернет все возможные комбинации мощностью n или менее

Permutations[{1,2,3},2]

{{},{1},{2},{3},{1,2},{1,3},{2,1},{2,3},{3,1},{3,2}}

Permutation[список, {n}] вернет все возможные комбинации мощность точно равной n

Permutations[{1,2,3},{2}]

{{1,2},{1,3},{2,1},{2,3},{3,1},{3,2}}

Поскольку функция **Permutation**[] как и другие функции перестановок работают со списками, при решении задач перестановок в области криптографии, необходимо разделять исходный текст на составляющие части. Для этого применяются функции **Characters**[] и **StringSplit**[].

Функция **Chararters**[] разделяет строку на составляющие ее символы

Characters["прилетаю седьмого вечером"]

{п,р,и,л,е,т,а,ю, ,с,е,д,ь,м,о,г,о, ,в,е,ч,е,р,о,м}

Функция **StringSplit**[], вызванная без дополнительных параметров разделяет строку на слова по пробелам.

StringSplit["прилетаю седьмого вечером"]

{прилетаю,седьмого,вечером}

Результатами работы функций **Characters**[] и **StringSplit**[] являются списки, которые можно применять как входные параметры функции **Permutation**[]

Permutations[StringSplit["прилетаю седьмого вечером"]]

{{прилетаю,седьмого,вечером}, {прилетаю,вечером,седьмого}, {седьмого,прилетаю,вечером}, {седьмого,вечером,прилетаю}, {вечером,прилетаю,седьмого}, {вечером,седьмого,прилетаю}}

Разумеется, в прикладной криптологии нет необходимости вычислять все возможные варианты перестановок. Часто возникает задача вычислить одну единственную перестановку по определенным правилам. В Wolfram Mathematica для этого служит функция **Permute**[].

Базовый синтаксис:

**Permute**[список,правило перестановки]

Со списком исходных элементов все понятно. В нашем случае это или список букв исходного текста, полученный с помощью функции **Characters**[], или список слов (**StringSplit**[]), или их комбинация.

Правило перестановок задается специальной функцией **Cycles**[]. Параметром этой функции является список указывающий на порядок перестановки. Например

**Cycles**[{{1,2}}]

Показывает, что первый элемент списка встанет на вторую позицию, а второй на первую 1->2->1.

Permute[{1,2,3,4,5},Cycles[{{1,2}}]]

{2,1,3,4,5}

Цикл **Cycles**[{{1,2,5}}] даст последовательность перестановок 1->2->5->1. Т.е первый элемент на вторую позицию, второй – на пятую, пятый на первую.

In[26]:= Permute[{1,2,3,4,5},Cycles[{{1,2,5}}]]

Out[26]= {5,1,3,4,2}

За один вызов перестановки можно осуществлять перестановку по нескольким циклам, например

Permute[{1,2,3,4,5},Cycles[{{1,2,5},{3,4}}]]

{5,1,4,3,2}

Здесь в дополнение к предыдущей перестановке поменяны местами 3 и 4 элементы.

Зная исходный массив и перестановку можно вычислить правило перестановки с помощью функции **FindPermutation[]**

FindPermutation[{5,1,4,3,2}]

Cycles[{{1,2,5},{3,4}}]

Или

FindPermutation[{1,2,3,4,5},{5,1,4,3,2}]

Cycles[{{1,2,5},{3,4}}]

Приложение 4. Генерация магических квадратов

<http://mathematica.stackexchange.com/questions/73131/the-magic-square-function> (переставить в литературу)

magic[n\_Integer /; (n > 0 && n != 2)] := Module[{m, j, k, p, i},

(\*Translation of Cleve Moler's magic magic() function to Mathematica\*)

Which[

Mod[n, 2] == 1, m = oddOrderMagicSquare[n],

Mod[n, 4] == 0,

j = Floor @ Abs [ Mod[Range[n], 4]/2];

k = Outer[Equal, j, j] /. {True -> 1, False -> 0};

m = Outer[Plus, Range[1, n\*n, n], Range[0, n - 1]];

p = Position[k, 1];

(m[[Sequence @@ #]] = n\*n + 1 - m[[Sequence @@ #]]) & /@ p,

True,

p = n/2;

m = oddOrderMagicSquare[p];

m = ArrayFlatten@{{m, m + 2\*p^2}, {m + 3\*p^2, m + p^2}};

If[n != 2,

i = Range[p];

k = (n - 2)/4;

j = {Range[k], Range[n - k + 2, n]};

j = Flatten@DeleteCases[j, {}];

m[[Join[i, i + p], j]] = m[[Join[i + p, i], j]]

]

];

m

];

oddOrderMagicSquare[n\_] := Module[{p},

p = Range[n];

Transpose[n\*Mod[Map[p + # &, p - (n + 3)/2], n] +

Mod[Map[p + # &, 2\*p - 2], n] + 1]

];