ЛАБОРАТОРНАЯ РАБОТА № 4

Тема работы: Стеганографические методы скрытия данных. Метод замены наименее значащего бита.

Цель работы: Познакомиться с одним из наиболее распространенных методов сокрытия информации в неподвижных изображениях.

Ключевые понятия, которые необходимо знать: стеганография, наименее значащий бит, RGB-палитра, форматы графических файлов, структура графического файла.

Время выполнения лабораторного занятия: 6 часов.

1. Теоретическая часть

Метод замены наименее значащего бита (НЗБ, LSB – Least Signification Bit) наиболее распространен среди методов замены в пространственной области.

Младший значащий бит изображения несет в себе меньше всего информации. Известно, что человек в большинстве случаев не способен заметить изменений в этом бите. Фактически, НЗБ – это шум, поэтому его можно использовать для встраивания информации путем замены менее значащих битов пикселей изображения битами секретного сообщения. При этом для изображения в градациях серого (каждый пиксель изображения кодируется одним байтом) объем встроенных данных может составлять 1/8 от общего объема контейнера. Например, в изображение размером 512. 512 можно встроить ~ 32 кБайт информации. Если же модифицировать два младших бита (что также практически незаметно), то данную пропускную способность можно увеличить вдвое.

Популярность данного метода обусловлена тем, что он позволяет скрывать в относительно небольших файлах достаточно большие объемы информации. Метод зачастую работает с растровыми изображениями, представленными в формате без компрессии (например, GIF и BMP).

Метод НЗБ имеет низкую стеганографическую стойкость к атакам пассивного и активного нарушителей. Основной его недостаток – высокая чувствительность к малейшим искажениям контейнера. Для ослабления этой чувствительности часто применяют помехоустойчивое кодирование.

2. Практическая часть

Рассмотрим пошаговую реализацию метода НЗБ в программе Mathcad. Вначале работы скопируйте изображение – контейнер «mushu.bmp» из папки преподавателя в папку, куда вы будете сохранять документ Mathcad.

Шаг 1. Импорт графического файла выполняется операцией **Picture** из позиции **Insert** главного меню программы. В модуле, который при этом появился, необходимо заполнить шаблон данных в левом нижнем углу, для чего в двойных кавычках следует ввести имя файла (или же, при необходимости, - полный путь его размещения на диске) и нажать клавишу <Enter>.

Пример цветного изображения, восстановленного с помощью операции **Picture** представлен на рис.1. Это изображение имеет размер 128×166 пикселей, глубина цвета – 24 бита. Для возможности обработки изображения необходимо перевести цветовые характеристики каждого его пикселя в числовую матрицу. Для выполнения этой операции применяется функция **READRGB** («имя файла»), возвращающая массив из трех подмассивов, которые в свою очередь, несут информацию о разложении цветного изображения на цветовые компоненты - красный (Red), зеленый (Green) и синий (Blue):

C:=READRGB("mushu.bmp")

"mushu.bmp"

Рис. 1. Изображение - контейнер

При этом три цветовых компонента размещаются один за другим в общем массиве ***C***. На рис. 2 представлена графическая интерпретация массива ***C*** в виде изображения с градациям серого. Образ слева характеризует интенсивность красного цвета в каждом пикселе изображения ***mushu.bmp***, средний – интенсивность зеленого цвета, справа – интенсивность синего цвета.

Рис.2. Графическая интерпретация массива цветовых компонентов контейнера-оригинала

Для выделения цветовых составляющих можно использовать встроенные функции выделения соответствующих цветовых компонентов, каждая из которых возвращает массив, соответствующий определенному цветовому компоненту графического файла:

R:=READ_RED("mushu.bmp")
G:=READ_GREEN((mushu.bmp")
B:=READ_BLUE("mushu.bmp")

Шаг 2. В качестве сообщения, которое необходимо срыть, можно использовать, например, первые несколько абзацев теоретической части к этой лабораторной работе. Текст сообщения сохраните в файле *prim.txt*, в той же папке, что и документ Mathcad, в следующем формате:

а) тип файла – обычный текст (*.txt);

б) кодировка текста – кириллица (Windows).

Примечание. *В принципе существует возможность скрытия файлов любого формата. Единственное условие, которое при этом должно выполняться, - выбор файла-контейнера надлежащего объема (ориентировочное соотношение между объемами файла-контейнера и файла-сообщения для метода НЗБ – 8:1).*

Импорт текстового сообщения можно выполнить с помощью функции **READBIN** («имя файла», «тип формата данных»). В нашем случае данные представлены как последовательность 8-битных беззнаковых целых чисел (байтов):

M:=READBIN("prim.txt""byte")

Результатом данного выражения является матрица-столбец (вектор), каждый элемент которого соответствует ASCII-коду соответствующего символа (буквы) импортируемого сообщения. В десятичном виде коды символов могут принимать значения от 0 до 255; в двоичном виде для этого достаточно использовать 8 битов на один символ – так называемое однобайтовое кодирование, на что и указывает параметр *byte* как аргумент функции **READBIN**.

Фрагмент импортированного сообщения, а именно коды первых 15 символов (включая пробелы) в десятичном и в двоичном видах, представлен на рис.3.

M =

	0
0	49
1	46
2	32
3	210
4	229
5	238
6	240
7	229
8	242
9	232
10	247
11	229
12	241
13	234
14	224
15	255
16	…

M =

	0
0	110001b
1	101110b
2	100000b
3	11010010b
4	11100101b
5	11101110b
6	11110000b
7	11100101b
8	11110010b
9	11101000b
10	11110111b
11	11100101b
12	11110001b
13	11101010b
14	11100000b
15	11111111b
16	…

Рис.3. Фрагмент сообщения, подлежащий скрытию

Примечание. *Для изменения формата отображения данных в Mathcad необходимо выполнить следующее: выбрать команду* ***Result*** *из меню* ***Format****. В появившемся диалоговом окне на вкладке* ***Display Options*** *в выпадающем меню* ***Radix*** *выбрать необходимую основу счисления –* ***binary****, а на вкладке* ***Number format*** *в списке* ***Format*** *выбрать позицию* ***Decimal****.*

Необходимо отметить, что по умолчанию нижняя граница индексации массивов равна 0.

Проверку импортирования файла сообщения (если в качестве сообщения используется обыкновенный текст) можно выполнить с помощью

вызова функции **vec2str(M)**. Эта функция возвращает строку символов, соответствующих вектору ***M*** ASCII-кодов.

Примечание. Возможна ситуация некорректного отображения кириллических символов в полученной строке.

Шаг 3. Перед скрытием текстового файла prim.txt в контейнере mushu.bmp его можно защитить криптографическим кодированием. Используем для этого модифицированный код Виженера.

Алфавит источника сообщения зададим в виде ASCII-кодов, значения которых, как известно, могут быть в диапазоне от 0 до 255. Зададим объем алфавита источника:

$$Na:=256$$

Из символов алфавита задаем секретный ключ, например:

$$K:="@J|eKc\text{-}I980"$$

Количество символов в ключе определим с помощью функции **strlen( )**:

$$Nk:=strlen(K)=11$$

Объем сообщения, которое подлежит кодированию:

$$Nm:=rows(M)=1.143\times10^{3}$$

где **rows(M)** – функция, которая возвращает количество строк массива ***M***, т.е. сообщение ***M*** содержит 1143 символа, включая скрытые служебные ASCII-символы переноса строки и возврата каретки.

Расширяем ключ на длину сообщения ***Nm***, используя программный модуль (М.1).

$$\begin{array}{l} K1:= \\ \\ \\ \\ \\ \\ \end{array} \left| \begin{array}{l} K \leftarrow str2vec(K) \\ \text{for } i \in 0..Nm-1 \\ \quad r \leftarrow mod(i, Nk) \\ \quad K1_i \leftarrow K_r \\ K1 \end{array} \right. \qquad (М.1)$$

В данном модуле использованы функции:

- **str2vec(K)** – преобразует строку символов ключа *K* в вектор их ASCII-кодов;

- **mod(i, Nk)** – возвращает остаток от деления *i* на *Nk*.

Выполняем кодирование сообщения, используя модуль (М.2).

$$M\_cod := \left| \begin{array}{l} \text{for } j \in 0..Nm-1 \\ \quad M\_cod_j \leftarrow \text{mod}\left(M_j + K1_j, Na\right) \\ M\_cod \end{array} \right. \qquad \text{(М.2)}$$

Шаг 4. Для того чтобы при распаковке контейнера из полученного множества символов можно было четко определить начало и конец именно скрытого сообщения, целесообразно ввести соответствующие секретные метки, которые ограничивали бы это полезное содержание.

Метки должны состоять из достаточного количества символов, чтобы не принимать за метки символы случайного образования. Кроме того, для уменьшения вероятности обнаружения меток при проведении стеганоанализа желательно, чтобы коды этих символов были достаточно разнесены на ASCII-оси (например, использовать наряду с латинскими символами символы кириллицы и служебных символов – так называемая транслитерация; использование псевдослучайных последовательностей кодов символов и т.п.). Пусть метки имеют следующий вид:

m_start:="n0G@m0k" m_end:="KiHeu,6"

Ограничивающие метки добавляем в текст кодированного сообщения для чего используем функцию **stack(A, B, ...)**, которая позволяет объединять записанные через запятую массивы. Объединяемые массивы должны иметь одинаковое количество столбцов, поэтому необходимо преобразовать метки из строк в векторы AS-CII-кодов. Следовательно, получим:

sMe:=stack(str2vec(m_start), M_cod, str2vec(m_end))

Общее количество символов в скрытом сообщении:

rows(sMe)=1157

Количество наименее значащих бит (НЗБ) контейнера, которое для этого необходимо (8 бит/символ):

8rows(sMe)=9256

Общее количество НЗБ контейнера:

Таким образом, файл изображения имеет достаточный объем для того, чтобы скрыть сообщение.

3 rows(C)·cols(C)=191232

Шаг 5. Для дальнейших вычислений потребуется преобразование десятичного числа (которым по умолчанию кодируется каждый символ) в формат двоичного. Также понадобится и обратное преобразование. Поскольку данные функции в Mathcad отсутствуют, создадим соответствующие функции с помощью модулей.

Преобразование двоичного числа x, которое задано матрицей столбцом (причем первый элемент матрицы – самый младший разряд числа) в десятичное выполняется с помощью модуля (М.3).

$$B2D(x) := \sum_{i=0}^{7} \left(x_i \cdot 2^i \right) \qquad \text{(М.3)}$$

Преобразование десятичного числа в двоичное реализуется модулем (М.4).

$$D2B(x) := \left| \begin{array}{l} \text{for } i \in 0..7 \\ \quad \left| \begin{array}{l} V_i \leftarrow \text{mod}(x, 2) \\ x \leftarrow \text{floor}\left(\frac{x}{2} \right) \end{array} \right. \\ V \end{array} \right. \qquad \text{(М.4)}$$

Функция **mod(x,2)** возвращает остаток от деления x на 2 (0, если x четное, и 1, если x нечетное). Функция **floor()** возвращает наибольшее целое число, которое меньше или равняется действительному значению аргумента.

Шаг 6. Для большего удобства и наглядности дальнейших действия развернем матрицу ***C*** в вектор, временно изменив порядок цветовых матриц с

R-G-B на B-G-R, что увеличит защищенность скрытой информации (на данном этапе можно использовать более надежные, но и более сложные алгоритмы). В нашем случае применим модуль (М.5), в котором функция **augment(A, B, …)** объединяет матрицы ***A, B, …,*** имеющие одинаковое количество строк.

$$\begin{aligned} Cv := \;& C1 \leftarrow \text{augment}(B, G, R) \\ & Cv \leftarrow C1^{\langle 0 \rangle} \\ & \text{for } i \in 1..\text{cols}(C1) - 1 \\ & \quad Cv \leftarrow \text{stack}(Cv, C1^{\langle i \rangle}) \end{aligned} \qquad \text{(М.5)}$$

Операция $C^{<I>}$ позволяет выбирать ***i***-й столбец из матрицы ***C***, каждый из которых впоследствии прибавляется к результирующему вектору ***Cv***. Функция **cols(C)** возвращает общее количество столбцов массива ***C***.

Шаг 7. На основе вектора **Cv** формируем новый вектор, который уже будет содержать скрытое закодированное сообщение (модуль М.6).

Каждый символ закодированного сообщения (операция цикла **for m∈0..rows(sMe)-1**) переводится в двоичный формат (переменная ***b***), каждый из восьми разрядов которого записывается вместо нулевого бита числа, соответствующего интенсивности того или иного цвета пикселя (последнее также предварительно переводится в двоичный формат (переменная **P**)).

После изменения модифицированное двоичное число **P** переводится в формат десятичного и записывается в соответствующую позицию вектора ***Sv***. (М.6)

После обработки последнего символа сообщения ***sMe*** выполняется модификация элементов массива ***Cv***, которые еще не претерпели изменений. Младшим битам каждого из таких элементов присваивается значение 0 или 1 (в данном случае по равномерному закону распределения; функция **round()** возвращает округленное до ближайшего целого значение аргумента), хотя более правильно было бы провести исследование закона распределения значений уже модифицированных младших битов и соответствующим образом изменять те, которые остались. Это делает невозможным со временем обнаружить факт модификации изображения.

В противном случае, проанализировав изображение, построенное из одних только НЗБ контейнера, нарушитель в большинстве случаев (если символов «не хватило» на весь контейнер) обнаружит границу ввода данных и при определенных усилиях сможет добыть скрытую информацию. Обычно эту информацию еще необходимо расшифровать, но факт ее наличия уже

будет раскрыт и вопрос защиты вернется к криптографической устойчивости используемого кодирования.

```
Sv := | for m ∈ 0..rows(sMe) − 1
      |   b ← D2B sMe_m
      |   for i ∈ 0..7
      |     P ← D2B Cv_(i+8·m)
      |     P_0 ← b_i
      |     Sv_(i+8·m) ← B2D(P)
      | for j ∈ rows(Sv)..rows(Cv) − 1
      |   P ← D2B Cv_j
      |   P_0 ← round(md(1))
      |   Sv_j ← B2D(P)
      | Sv
```

(М.6)

Желательно, чтобы изображение, которое планируется использовать в качестве контейнера, было уникальным. Следует заметить, что все графические контейнеры условно разделяются на «чистые» и «зашумленные». У первых прослеживается связь между младшими и остальными семью битами цветовых компонентов, а также зависимость между самыми младшими битами.

Встраивание сообщения в «чистое» изображение разрушает существующие зависимости, что легко обнаруживается. Если же изображение изначально зашумлено (сканированное изображение, фото и т.д.), то определить постороннее вложение становится на порядок сложнее, хотя и возможно при использовании теории вероятностей и математической статистики.

Также можно отметить, что для большей скрытности биты сообщения следует вносить не последовательно, а только в каждый второй или даже третий пиксель, или же подчинить внесение определенному, известному только авторизованным лицам закону. Данные модификации легко осуществить путем внесения соответствующих незначительных изменений в модуль (М.6).

Шаг 8. Полученный с помощью модуля (М.6) вектор **Sv** сворачиваем в матрицу ***S***, имеющую размерность первичной матрицы ***C*** (модули (М.7) и (М.8)).

S1:=for i∈0cols(C)−1

$$S1^{l} \leftarrow \text{submatrix}[Sv, i\cdot rows(C), (i+1)\cdot rows(C)-1, 0, 0] \qquad (M.7)$$

Функция **submatrix(A,x,X,y,Y)** возвращает часть матрицы ***A***, которая состоит из элементов, общих для строк от ***x*** до ***X*** и столбцов от ***y*** до ***Y*** включительно.

Шаг 9. Пользуясь этой же функцией, выделяем из массива ***S*** цветовые матрицы и расставляем их на свои места (R↔B), получая контейнер-результат ***S*** (М.8).

$$B\_M := \text{submatrix}\left(S1, 0, rows(C)-1, 0, \frac{cols(C)}{3}-1\right)$$

$$G\_M := \text{submatrix}\left(S1, 0, rows(C)-1, \frac{cols(C)}{3}, 2\frac{cols(C)}{3}-1\right)$$

$$R\_M := \text{submatrix}\left(S1, 0, rows(C)-1, 2\frac{cols(C)}{3}, cols(C)-1\right)$$

$$S := \text{augment}(R\_M, G\_M, B\_M)$$

(М.8)

На рис.4 показана графическая интерпретация массива ***S*** в виде изображения с градациями серого и воспроизведенное по цветовым составляющим изображение-контейнер со скрытым сообщением.

Сравнивая рис.4 с рис.1. можно сделать вывод об отсутствии заметных визуальных отклонений.

R_M, G_M, B_M S

Рис. 4. Контейнер-результат и его интерпретация в виде массива цветовых компонентов

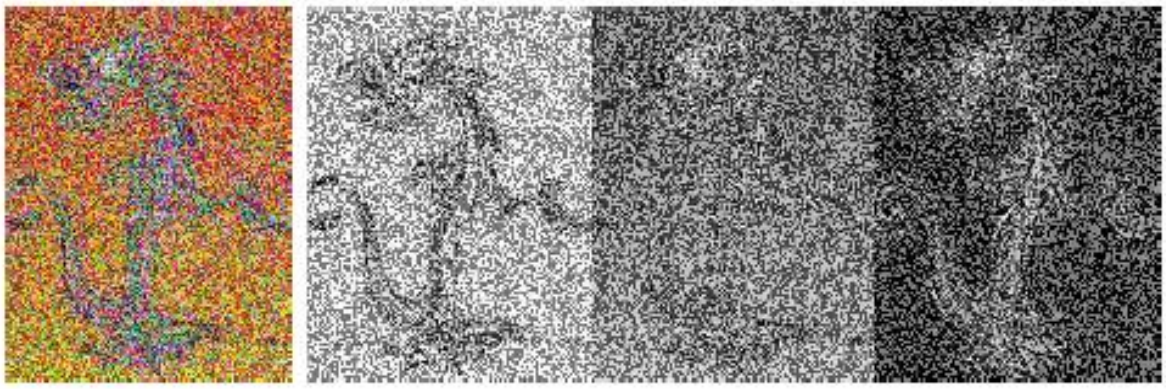
Остается только записать массив ***S*** в файл:

WRITERGB("mushu1.bmp"):=S

Совершенно очевидно, что объем полученного файла будет соответствовать объему фала изображения-оригинала.

Шаг 10. Для исследования влияния на степень скрытости того, в какой из разрядов числа, характеризующего то или иное свойство пикселя (в нашем случае – интенсивность цвета), будет заноситься информация, в модуле (М.6) в двух строках, вида $P_0 \leftarrow \ldots$, следует вместо индекса «0» ввести индекс, соответствующий модифицируемому разряду. На рис.5 изображен результат, полученный при внесении данных в 7-й (самый старший) бит числа, соответствующего интенсивности цветов.

Установлено, что визуально незаметно, если в качестве «носителей» использовать не только самый младший, но и следующий за ним бит каждого из указанных выше чисел (особенно если в изображении отсутствуют большие однотонные участки). Следовательно, в качестве одной из возможных степеней защиты возможно использование изменяемой по определенному закону поочередной записи в эти два бита. Или же жертвуя скрытностью, можно вдвое увеличить емкость контейнера.



R_M7, G_M7, B_M7 S7

Рис. 5. Цветное изображение и массив цветовых составляющих в случае внесения скрываемых данных в 7-й бит интенсивности цветов

Шаг 11. Рассмотрим процесс распаковки скрытого сообщения. Для этого создадим новый документ Mathcad. Скопируем в новый документ модули (М.3) и (М.4), осуществляющие перевод числа из десятичной системы счисления в двоичную и наоборот. Изначально зная, что сообщение было помещено в массив цветовых компонентов, выделим соответствующие каждому цвету подмассивы, переводя значения цветовых характеристик каждого пикселя изображения, содержащего в себе закрытое закодированное сообщение, в числовые матрицы:

R1:=READ_RED("mushu1.bmp")

G1:=READ_GREEN("mushu1.bmp")

B1:=READ_BLUE("mushu1.bmp")

Из полученных матриц соответствующим образом меняя их порядок, аналогично (М.5) формируем вектор ***Sv1*** (модуль (М.9)).

$$\begin{aligned} Sv1 := \;& S1 \leftarrow \text{augment}(B1, G1, R1) \\ & Sv1 \leftarrow S1^{\langle 0 \rangle} \\ & \text{for } i \in 1..\text{cols}(S1) - 1 \\ & \quad Sv1 \leftarrow \text{stack}\left(Sv1, S1^{\langle i \rangle}\right) \end{aligned}$$

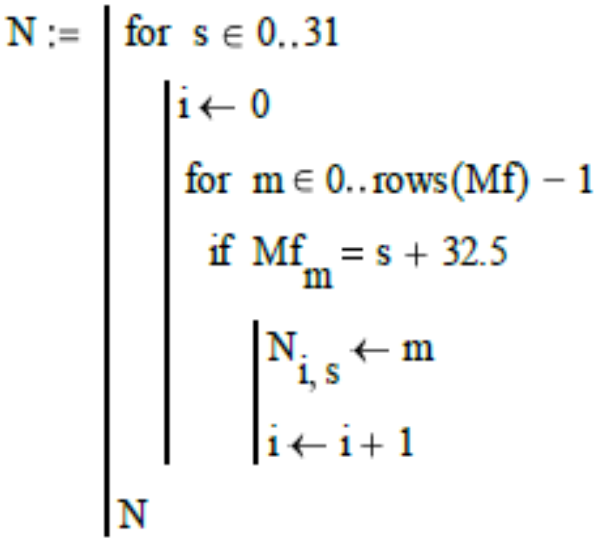
(М.9)

Надлежащим образом обрабатывая каждые восемь элементов полученного вектора, распаковываем скрытое сообщение, используя модуль (М.10).

$$
\begin{aligned}
Mf := \;& \text{for } m \in 0..\frac{\text{rows}(Sv1)}{8} - 1 \\
& \quad \text{for } i \in 0..7 \\
& \quad\quad P \leftarrow D2B\left(Sv1_{i+8\cdot m}\right) \\
& \quad\quad b_i \leftarrow P_0 \\
& \quad Mf_m \leftarrow B2D(b) \\
& \quad Mf_m \leftarrow Mf_m + 32.5 \text{ if } Mf_m < 32 \\
& Mf
\end{aligned}
\tag{М.10}
$$

Следует отметить, что поскольку заранее неизвестно какую часть вектора ***Sv1*** займет именно полезная информация, во внимание берутся все его элементы. Значения каждого элемента формируемого при этом вектора ***Mf*** представляют собой коды символов «квазисообщения», которые вычисляются в обратном к (М.6) порядке: каждый младший разряд восьмерки преобразованных в двоичный формат элементов вектора ***Sv1*** формирует двоичное число кода символа, формат которого далее преобразуется в десятичный. Полученное число присваивается ***m***-му элементу вектора ***Mf***.

Шаг 12. В связи с невозможностью обработки строковыми функциями Mathcad символов, ASCII-коды которых имеют значения от 0 до 31 включительно (за исключением служебных символов LF (код 10) и CR (код 13)), дополнительно вносится замена значений 0, 1, 2, …, 31 элементов вектора ***Mf*** соответствующим прибавлением к каждому из них коэффициента 32,5 (коэффициент является дробью для того, чтобы в дальнейшем было возможно отличить «настоящие» значения элементов массива от тех, которые перед этим имели неформатные значения). Такую замену, конечно, необходимо соответствующим образом учитывать в дальнейшем – для этого запомним номера строк вектора ***Mf***, элементы которых имеют дробные значения (модуль (М.11)).



(М.11)

При этом в первый столбец формируемого массива ***N*** заносятся номера элементов, значения которых равны 32,5 (бывшие нули), во второй столбец – номера элементов со значением 33,5 (бывшие единицы) и т.д.

Шаг 13. Зная, что текст полезной информации ограничен метками

m1_start:="n0G@m0k" m1_end:="KiHeu,6"

выделяем его из извлеченного квазисообщения, используя модуль (М.12).

Вектор ASCII-кодов ***Mf***, предварительно преобразованный с помощью функции ***vec2str()*** в соответствующую ему строку символов, последовательно просматривается в поиске стартовой и конечной меток. Такая операция выполняется путем сравнения выделенной части строки данных с ответствующими метками, которые должны быть известны получателю.

Выделение выполняется с помощью функции **substr(Mf, m, b),** которая возвращает подстроку длиной ***b*** символов из строки ***Mf***, начиная с символа ***m*** (следует отметить, что по умолчанию первый символ строки имеет номер 0). В нашем случае, поскольку каждая из меток состоит из 7 символов, b=7.

Последовательным увеличением ***m***, происходит продвижение вдоль строки данных ***Mf***. При выполнении указанных в модуле условий (оператор ***if***), коэффициентам ***s*** и ***e*** присваиваются соответствующие значения номеров начальной и конечной позиции полезной информации в строке данных ***Mf.*** Такие дополнительные условия, как ***s=0***; ***e=0***; s≠0; e≠0 введены для ускорения поиска.

Обратное преобразование строки символов ***Mf*** в вектор их ASCII-кодов позволяет в этом же модуле непосредственно выделить скрытую

информацию и восстановить элементы, значения которых были принудительно изменены на дробные.

```
M_cod1 := | s ← 0
          | e ← 0
          | bs ← strlen(m1_start)
          | be ← strlen(m1_end)
          | Mf ← vec2str(Mf)
          | for m ∈ 0..strlen(Mf) − 1
          |   | s ← m + bs  if substr(Mf, m, bs) = m1_start ∧ s = 0
          |   | e ← m − 1  if substr(Mf, m, be) = m1_end ∧ e = 0
          |   | break  if s ≠ 0 ∧ e ≠ 0
          | Mf ← substr(Mf, s, e − s + 1)
          | M_cod1 ← str2vec(Mf)
          | for n ∈ 0..cols(N) − 1
          |   for i ∈ 0..rows(N) − 1
          |     | break  if N[i,n] = 0 ∨ N[i,n] > e
          |     | M_cod1[N[i,n] − bs] ← n  if s < N[i,n] ≤ e
          | M_cod1
```

(M.12)

Шаг 14. Распакованное сообщение требуется декодировать. Необходимые начальные условия состоят в том, что авторизованной стороне известны:

- алфавит источника сообщения (ASCII-символы) объемом Na256 символов;
- секретный ключ K:="@J|eKc-I980" из Nk:=strlen(K)=11 символов;
- объем сообщения, подлежащего декодированию: Nm:=rows(M_cod1)=1143 символов.

С помощью модуля (М.13) секретный ключ ***K*** расширяется на длину ***Nm*** сообщения ***M_cod1*** (аналогично тому, как это было при кодировании сообщения в модуле (М.1)).

```
K11 := | K ← str2vec(K)
       | for i ∈ 0..Nm − 1
       |     r ← mod(i, Nk)
       |     K1_i ← K_r
       | K1
```

(М.13)

Декодирование секретного сообщения выполняется с помощью модуля (М.14).

```
M1 := | for j ∈ 0..Nm − 1
      |   M_j ← mod(Na + M_cod1_j − K11_j, Na)
      | M
```

(М.14)

Декодированное сообщение записываем в файл:

WRITEBIN("M_dec.txt", "byte",0):=M1

3. Задание на лабораторную работу

Разработать в программе Mathcad реализацию метода замены наименее значащего бита в соответствии с шагами, описанными в практической части методических указаний.

4. Контрольные вопросы

1. Какая цель методов стеганографии?
2. Как выполняется сокрытие данных методом наименее значащего бита?
3. Как зависит визуальное качество изображения-контейнера от номера использованного бита (битов) для скрытия данных по методу НЗБ?
4. Какие данные можно скрыть методом НЗБ?
5. Как определить максимальный объем данных, которые можно скрыть в файле-контейнере по методу НЗБ?
6. Как можно повысить скрытность данных в методе НЗБ?