Содержание

Лабораторная работа № 1 2
Метод рассечения-разнесения данных 2
Задание 4
Лабораторная работа № 2 6
Зашитное кодирование по методу Хэмминга 6
Задание 8
Лабораторная работа № 3 11
Генерация псевлослучайных чисел 11
Задание 12
Лабораторная работа № 4 15
Шифрование методом гаммирования 15
Задание 16
Сроки слачи лабораторных работ 17

Лабораторная работа № 1

Метод рассечения-разнесения данных

В тех информационных системах, где хранимая информация (данные) размещается в файлах, для обеспечения конфиденциальности, помимо шифрования может использоваться метод рассечения-разнесения.

Суть метода рассечения-разнесения состоит в том, что набор защищаемых данных разбивается на блоки, которые разносятся по нескольким другим наборам данных. Каждый отдельный блок не несет сколько-нибудь значимой информации, и даже доступ к полной совокупности блоков не позволяет легко восстановить исходный набор данных без знания способа разбиения.

Пример использования метода рассечения-разнесения. Пусть защищается текст (символ «я» соответствует пробелу), который требуется разбить на 8 блоков: МЕТОДяРАССЕЧЕНИЯ-РАЗНЕСЕНИЯ.

Представим защищаемый текст в виде таблицы, состоящей из четырех столбцов. Для разбиения требуется выбрать два ключа – для столбцов и для строк. Столбцам таблицы сопоставляется ключ, состоящий из натуральных чисел, равных номерам столбцов (нумерация начинается с единицы), расположенных в случайном порядке. Поскольку в нашей таблице четыре столбца с номерами от 1 до 4, в качестве ключа можно взять последовательность {4-1-3-2}.

Длина ключа, соответствующего строкам, должна быть такой, чтобы произведение длин ключей столбцов и строк равнялось количеству блоков, на которые разбивается текст. В нашем случае –

двум, например, {2-1}. Сопоставим этот ключ каждой паре строк таблицы.

Ключи	***4***	***1***	***3***	***2***
2	М	Е	Т	О
1	Д	я	Р	А
2	С	С	Е	Ч
1	Е	Н	И	Я
2	-	Р	А	З
1	Н	Е	С	Е
2	Н	И	Я	**.**

Теперь, если обозначить через $\mathbf{r_i}$ значение i-й позиции ключа строки, через $\mathbf{s_j}$ – значение j-й позиции ключа столбца, а через **n** – число столбцов, то номер блока **K**, в который помещается очередной символ открытого текста, определяется значением выражения:

$$\mathbf{K = n\,(r_i - 1) + s_j} \qquad (*)$$

В соответствии с заданным правилом, первый символ текста «М» запишется в блок с номером **K**=4*(2-1)+4=8. Следующий символ «Е» попадет в блок с номером **K**=4*(2-1)+1=5. Третий символ «Т» – в блок с номером **K**=4*(2-1)+3=7. И так далее до конца текста.

Сформированные блоки будут иметь следующее содержимое:

Номер блока	Содержимое
1	яНЕ
2	АЯЕ
3	РИС

4	ДЕН
5	ЕСРИ
6	ОЧЗ.
7	ТЕАЯ
8	МС-Н

Таким образом, открытый текст заменяется восемью блоками, длина которых в сумме даст длину исходного текста.

При восстановлении исходного текста, по формуле (*) вычисляется номер блока, из которого извлекается очередной символ.

Задание

Составьте программу, которая будет выполнять процедуру рассечения-разнесения и обратную операцию – сборку, для строки текста произвольной длины, содержащего символы переноса строки (состоящего из нескольких абзацев). Разделяемыми элементами являются символы.

Программа должна обеспечивать удобный пользовательский интерфейс, предоставляя пользователю возможность ввода:

1. вида выполняемой операции (разбиение/сборка),
2. текстовой строки для разбиения,
3. ключей столбцов (в произвольной комбинации)
4. ключей строк (в произвольной комбинации).

Все вводимые и выводимые данные должны сопровождаться четкими и ясными для пользователя пояснениями.

Количество блоков, на которые разбивается исходный файл и длина ключа, соответствующего количеству столбцов, выбираются из

представленной далее таблицы. Номер варианта соответствует последней цифре номера студенческого билета.

Номер варианта	Количество блоков	Количество столбцов
0	10	5
1	12	4
2	15	5
3	9	3
4	16	4
5	8	4
6	14	2
7	12	3
8	10	2
9	8	2

Отчет о выполнении лабораторной работы должен включать:

1. Титульный лист.
2. Краткое описание метода.
3. Значения параметров разбиения (количество блоков, количество столбцов, длина ключей) согласно варианту.
4. Пример разбиения произвольной строки текста (не менее 20 символов) в соответствии с заданными вариантом параметрами и с произвольно выбранным ключом.
5. Фрагменты программы, выполняющие разбиение и слияние блоков,

сопровождаемые комментариями, с предварительным описанием основных использованных переменных и массивов (тип, размерность, назначение и т.п.). Каждый из этих фрагментов должен быть выполнен в виде одной функции (процедуры).

Работоспособность программы проверяется преподавателем.

Самоконтроль

1. Проверьте работоспособность программы на текстовом фрагменте, длина которого меньше, чем количество блоков в соответствии с вариантом.
2. Проверьте работоспособность программы для текстового фрагмента, состоящего более чем из одного абзаца.
3. Удостоверьтесь, что пользователь не может осуществить ввод значений ключей, содержащих:
    a. буквы;
    b. отрицательные числа;
    c. нуль;
    d. количество элементов большее, чем предельная длина ключа (например, для ключа из трех элементов недопустимой будет являться последовательность {1, 2, 3, 4}).
4. Убедитесь, что итогом осуществления последовательного разбиения текста с некоторым ключом и последующей сборки с тем же самым ключом, является исходный текст.
5. Убедитесь, что в том случае, если разбиение и сборка велись с разными

ключами, исходный текст в результате сборки не восстанавливается.

Лабораторная работа № 2

Защитное кодирование по методу Хэмминга

При хранении данных и передаче их по каналам связи, возможно возникновение ошибок, даже одна из которых способна существенно исказить смысл сообщения. Для обнаружения и устранения ошибок (восстановления исходного вида данных) применяются методы защитного кодирования, связанные с внесением в исходные данные некоторой избыточной информации.

Одним из методов, позволяющих фиксировать и исправлять единичные ошибки в блоке двоичных данных, является метод Хэмминга. При использовании этого метода вся последовательность данных разбивается на блоки фиксированной длины n. При кодировании длина фиксированного блока увеличивается до такой длины N, чтобы дополнительные биты позволяли сформировать число от нуля до N – для фиксации места ошибки (позиции ошибочного бита блока).

Метод требует формирования матрицы контрольного суммирования, каждый столбец которой представляет собой двоичное значение порядкового номера столбца (число столбцов равно N). Эта матрица будет умножаться (по модулю 2) на передаваемый вектор.

Битами исходной последовательности заполняется, в порядке их следования, передаваемая последовательность (вектор) длины N, исключая резервные позиции, соответствующие степеням числа 2. Резервные позиции формируются так, чтобы произведение каждой строки матрицы на этот вектор давало значение 0.

Проиллюстрируем формирование матрицы на примере

последовательности из десяти битов (n=10):

0 0 1 1 1 0 1 0 1 0.

Длина N новой последовательности должна быть не менее 14, так как резервные 4 позиции позволяют получить двоичное представление чисел от 0 до 15, что достаточно для отображения любого возможного места ошибки.

При описании матрицы контрольного суммирования использованы следующие обозначения:

i – номер бита в исходном сообщении;

j – номер столбца матрицы;

a_k – обозначение k-й строки матрицы;

b – биты исходного сообщения (резервные позиции пусты);

β – значения битов на резервных позициях, обеспечивающие нулевые значения вектора контрольных сумм;

b_t – передаваемый вектор ($b_t=b+\beta$);

b_r – пример вектора, переданного с ошибкой (в пятой позиции);

S_0 – вектор контрольных сумм, соответствующий исходной последовательности (с нулевыми значениями резервных разрядов);

S_t – вектор контрольных сумм, соответствующий передаваемой последовательности (со сформированными значениями резервных разрядов);

S_r – вектор контрольных сумм, соответствующий последовательности, переданной с ошибкой (в пятом бите).

i			1		2	3	4		5	6	7	8	9	10	S_0	S_t	S_r
j	1	2	3	4	5	6	7	8	9	10	11	12	13	14			

a_1	1	0	1	0	1	0	1	0	1	0	1	0	1	0	0	0	1
a_2	0	1	1	0	0	1	1	0	0	1	1	0	0	1	1	0	0
a_3	0	0	0	1	1	1	1	0	0	0	0	1	1	1	1	0	1
a_4	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	0	0
b			0		0	1	1		1	0	1	0	1	0			
β	0	1		1				1									
b_t	0	1	0	1	0	1	1	1	1	0	1	0	1	0			
b_r	0	1	0	1	1	1	1	1	1	0	1	0	1	0			

Задание

Составьте программу, которая будет выполнять кодирование методом Хэмминга произвольной битовой последовательности заданной вариантом длины (задается в виде последовательности символов 0 и 1, вводимых без пробелов). Длина последовательности выбирается из представленной ниже таблицы. Номер варианта соответствует последней цифре номера студенческого билета.

Номер варианта	Длина кодируемой последовательности
0	6
1	9
2	4
3	12
4	7
5	10

6	8
7	11
8	13
9	5

Исходная двоичная последовательность вводится пользователем по запросу программы, после чего программа формирует матрицу контрольного суммирования и другие данные так, как это показано в примере выше (кроме вектора контрольных сумм S_r) и отображает их на экране дисплея. Затем программа запрашивает **передаваемую** последовательность, вводимую пользователем, возможно, с одной ошибкой. После этого формируется вектор контрольных сумм S_r и сообщается номер позиции ошибки, если она есть.

Программа должна обеспечивать удобный пользовательский интерфейс. Все вводимые и выводимые данные должны сопровождаться четкими и ясными для пользователя пояснениями.

Отчет о выполнении лабораторной работы должен включать:

1. Титульный лист.
2. Значение длины кодируемого блока (до кодирования и после него) согласно варианту.
3. Пример кодирования первых битов произвольной, но при этом отличной от примера двоичной последовательности заданной вариантом длиной (должны быть приведены матрица контрольного суммирования, векторы, пояснения).

4. Фрагменты программы, связанные с выполнением основных операций (формирование матрицы контрольного суммирования, передаваемого вектора, векторов контрольных сумм), сопровождаемый комментариями, с предварительным описанием основных использованных переменных и массивов (тип, размерность, назначение и т.п.). Фрагменты должны быть выполнены в виде одной или нескольких функций (процедур).

Работоспособность программы проверяется преподавателем.

Самоконтроль

1. Удостоверьтесь, что поля для ввода исходной последовательности и передаваемой последовательности с ошибкой допускают ввод только последовательностей, состоящих из нулей и единиц, а также адекватно обрабатывают случаи ввода "слишком длинных" и "слишком коротких" последовательностей (т.е. последовательностей, длина которых меньше или больше той, что должна быть в соответствии с вариантом).
2. Убедитесь, что матрица контрольного суммирования строится адекватным образом для различных исходных последовательностей (в том числе и для последовательности, состоящей из одних нулей).
3. Проверьте работоспособность программы и адекватность номеров позиций ошибки в случаях:
    a. Наличия ошибки в первой позиции последовательности.
    b. Наличия ошибки в последней позиции последовательности.
    c. Отсутствия ошибок.
    d. Более чем одной ошибки в передаваемой последовательности.

Лабораторная работа № 3

Генерация псевдослучайных чисел

Во многих случаях, например, при создании ключей шифрования, требуется получить некоторое случайное число, которое и будет ключом. Для получения последовательности случайных (точнее, псевдослучайных, то есть периодически повторяющихся) чисел используются генераторы псевдослучайных чисел (ГПСЧ). В качестве примера простейшего ГПСЧ можно привести линейный конгруэнтный генератор, использующий рекуррентное соотношение вида:

$$\mathbf{c}_i = (\mathbf{a}*\mathbf{c}_{i-1} + \mathbf{b}) \pmod{\mathbf{m}}$$

где

$\mathbf{c}_i$ − i-е число псевдослучайной последовательности;

$\mathbf{c}_{i-1}$ − предшествующее число последовательности;

$\mathbf{c}_0$ − порождающее число последовательности;

a,**b**,**m** – натуральные числа (константы), удовлетворяющие определенным требованиям.

Значение **m** берется равным 2^n (где **n** – целое число). Период повторения псевдослучайных чисел зависит от выбираемых значений всех параметров, но никогда не превышает значения **m**.

Значение **b** должно быть нечетным числом, взаимно простым с **m**, а коэффициент **a** должен быть таким, чтобы **a** mod 4=1 (mod – остаток при делении с остатком). Значение $\mathbf{c}_0$ выбирается случайным образом. Значения **a**, **b** и $\mathbf{c}_0$ должны быть менее **m**.

Задание

Составьте программу, реализующую линейный конгруэнтный ГПСЧ при **n**=24 (то есть **m=**2^{24}), и способную генерировать случайные числа как по одному (с выводом на экран), так и последовательностью произвольной длины (с записью в файл). Значения параметров **a**, **b** и $\mathbf{c_0}$ должны генерироваться случайным образом, но без использования стандартного генератора случайных чисел в выбранном вами языке программирования. Необходимо придумать и реализовать способ получения случайных чисел в зависимости от времени суток, текущего положения курсора мыши или иных придуманных вами параметров. Если сгенерированное таким образом значение параметра не удовлетворяет требованиям, оно может каким-либо способом дополнительно изменяться.

Значения, генерируемые ГПСЧ должны иметь равномерное распределение. Для проверки этого, весь диапазон возможных значений ГПСЧ [0, **m**–1] делится на 100 интервалов равной длины (с округлением длины до целого, длина последнего интервала может отличаться от длины остальных). Во время или после генерации чисел подсчитываются относительные частоты попадания сгенерированных значений в каждый из интервалов (отношение числа значений, попавших в интервал, к общему числу сгенерированных значений). Затем вычисляется среднее арифметическое от относительных частот (в идеальном случае это будет 0.01 или 1%, то есть в каждый из ста интервалов попадет одна сотая от всех сгенерированных значений). Относительные частоты округляются либо минимум до 4 знаков после запятой (0.0079) в случае работы с

числовыми значениями, либо минимум до двух в случае работы с процентными (0,79%).

Итого, от программы требуется:

1) По запросу пользователя генерировать новые параметры ГПСЧ.

2) На основании текущих параметров ГПСЧ генерировать и выводить на экран следующее псевдослучайное число (столько раз, сколько понадобится пользователю).

3) На основании текущих параметров ГПСЧ сгенерировать последовательность псевдослучайных чисел заданной пользователем длины, записать её в файл, а на экран вывести гистограмму относительных частот попаданий сгенерированных значений в сто интервалов. При этом первое число генерируемой последовательности всегда должно рассчитываться на основании порождающего числа последовательности $\mathbf{c_0}$.

Пояснение к реализации пунктов 2 и 3 – если пользователь просмотрит по одному первые 10 чисел псевдослучайной последовательности на экране (пункт 2), а затем сгенерирует последовательность из 10 чисел (пункт 3), то эти последовательности совпадут. Далее, если пользователь сгенерирует последовательность из 15 чисел (снова пункт 3), то первые десять в ней совпадут с прошлой последовательностью. А если после этого пользователь попросит вывести следующее число на экран (пункт 2), то увидит одиннадцатое число из второй последовательности. Разумеется, при смене параметров ГПСЧ, должна изменяться и генерируемая последовательность.

Инициализирующая (генерация значений параметров ГПСЧ) и генерирующая части программы должны быть выполнены в виде

отдельных функций (процедур).

Отчет о выполнении лабораторной работы должен включать:

1. Титульный лист.
2. Описание способа генерации значений параметров **a**, **b** и $\mathbf{c_0}$ (и способ генерации случайных чисел, и действия в случае, когда сгенерированные числа не удовлетворяют требованиям к параметрам).
3. ***СТАТИСТИКА:*** Результаты проверки равномерности распределения линейного конгруэнтного ГПСЧ – средние, максимальные и минимальные относительные частоты попаданий в интервал, с указанием тех наборов значений **a**, **b** и $\mathbf{c_0}$, с которыми выполнялась проверка, а также длинами генерируемых последовательностей).
4. Текст функции (процедуры), выполняющей генерацию параметров **a**, **b** и $\mathbf{c_0}$.
5. Текст функции (процедуры), реализующей линейный конгруэнтный ГПСЧ.

Программа должна обеспечивать удобный пользовательский интерфейс. Все вводимые и выводимые данные должны сопровождаться четкими и ясными для пользователя пояснениями.

Работоспособность программы проверяется преподавателем.

Самоконтроль

1. Задайте длину последовательности равную 10 элементам. Визуально проверьте адекватность построения гистограммы. В идеальном случае она будет содержать ровно 10 столбцов "единичной" высоты и 90 "нулевых" столбцов. Вариант с наличием одного интервала, в который

попали два элемента, также является теоретически возможным и допустимым. Наличие более чем одного интервала с двумя попаданиями или появление интервала, в который попали три и более элементов является признаком ошибочной реализации ГПСЧ.

2. В случае 100 элементов максимальное количество попаданий в один интервал равно трем. Большее количество попаданий снова свидетельствует об ошибках в реализации ГПСЧ.

3. **HINT:** Однократное и не повторяющееся при перезапусках программы (с другими значениями **a**, **b** и $\mathbf{c_0}$) наблюдение чрезмерно большого количества попаданий в какой-либо интервал вполне допустимо, так как оно является побочным эффектом несовершенства процедуры генерации **a**, **b** и $\mathbf{c_0}$. Если ситуация с "чрезмерным количеством попаданий" воспроизводится регулярно, следует изменить способ генерации **a**, **b** и $\mathbf{c_0}$ и повторить тестирование.

Лабораторная работа № 4

Шифрование произвольного файла методом гаммирования

При передаче в сетях больших объемов данных для шифрования используется метод гаммирования (разновидность одноразовой системы шифрования), связанный с наложением на исходное сообщение, рассматриваемое как последовательность битов, некоторой секретной гаммы (ключа шифрования) – псевдослучайной битовой последовательности той же длины.

Над каждой парой битов с одинаковыми порядковыми номерами из исходной и псевдослучайной последовательностями выполняется операция суммирования по модулю два (XOR или исключающее ИЛИ – E):

$$\mathbf{Y}_i = \mathbf{X}_i \text{ E } \mathbf{K}_i$$

где

i – порядковый номер бита;

$\mathbf{Y}_i$ – i-й бит шифрованного сообщения;

$\mathbf{X}_i$ – i-й бит исходного сообщения;

$\mathbf{K}_i$ – i-й бит псевдослучайной последовательности.

Расшифрование осуществляется путем наложения той же ключевой гаммы на шифрованное сообщение и выполнении той же операции:

$$\mathbf{X}_i = \mathbf{Y}_i \text{ E } \mathbf{K}_i$$

Для получения гаммы используются генераторы псевдослучайных чисел (ГПСЧ), например, линейный конгруэнтный генератор.

Задание

Составьте программу, которая будет:

1. Генерировать значения параметров **a**, **b** и $\mathbf{c_0}$ для линейного конгруэнтного ГПСЧ и сохранять их в указываемом пользователем файле-ключе со стандартным расширением "key". Файл может быть как новым, так и заменить уже существующий – с предварительным подтверждением замены.
2. Выполнять шифрование гаммированием выбранного пользователем файла с использованием файла-ключа, который тоже указывается пользователем. Последовательность чисел для гаммы генерируется линейным конгруэнтным ГПСЧ на основе параметров в файле-ключе, длина последовательности зависит от длины шифруемого файла. При гаммировании должны использоваться все биты каждого случайного числа. Результат шифрования записывается в указываемый пользователем файл, который либо создается в случае отсутствия, либо заменяет существующий с предварительным подтверждением замены в случае существования (Для самопроверки: если выполнить гаммирование зашифрованного файла с тем же файлом-ключом, то получится исходный файл).
3. Выводить на экран время выполнения гаммирования файла.

Программа должна обеспечивать удобный пользовательский интерфейс. Все вводимые и выводимые данные должны сопровождаться четкими и ясными для пользователя пояснениями.

Отчет о выполнении лабораторной работы включает:

1. Титульный лист.
2. Описание структуры файла-ключа (способ хранения в нем параметров ГПСЧ).
3. ***СТАТИСТИКА:*** Анализ зависимости времени гаммирования от размера файла (на основании минимум 10 различных измерений).
4. Фрагмент программы (функция или процедура), выполняющий гаммирование, сопровождаемый комментариями, с предварительным описанием основных использованных переменных и массивов (тип, размерность, назначение и т.п.).

Работоспособность программ проверяется преподавателем.

Самоконтроль

1. Проанализируйте результаты сбора статистики. Убедитесь, что зависимость между размером файла и временем, требующимся на его гаммирование, является близкой к линейной (т.е. пот возрастании размера файла в 2 раза, время гаммирования возрастает приблизительно в 2 раза).
2. В том случае, если зависимость между размером файла и временем гаммирования заметно отличается от линейной, убедитесь, что время, засекаемое программой, включает в себя только время гаммирования, без учета времени, требуемого для чтения или записи информации в используемые файлы.

Сроки сдачи лабораторных работ

Номер лабораторной	Срок
1	20.10
2	03.11
3	24.11
4	22.12