Министерство образования и науки Российской Федерации

Федеральное государственное бюджетное образовательное учреждение
высшего профессионального образования

**ТОМСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ СИСТЕМ
УПРАВЛЕНИЯ И РАДИОЭЛЕКТРОНИКИ (ТУСУР)**

Кафедра автоматизированных систем управления (АСУ)

А.Я. Суханов

Информатика

Учебное методическое пособие по практическим занятиям,
самостоятельной и индивидуальной работе студентов направления
09.03.03 Прикладная информатика в экономике
09.03.01 Информатика и вычислительная техника

Томск 2020

Суханов А.Я.

Информатика: Учебное методическое пособие по практическим занятиям, самостоятельной и индивидуальной работе студентов – 37 с.

Учебное методическое пособие содержит программу и задания для проведения и выполнения практических занятий.

Оглавление

Введение 4
1. Практическое занятие: Кодирование, представления числовой информации 5
1.1 Равномерное и неравномерное кодирование 5
1.1.1 Кодирование Шеннона Фано 5
1.1.2 Кодирование Хаффмана 9
1.2 Кодирование в условиях помех 11
1.2.1 Блочные коды, код Хэмминга 14
1.2.2 Сверточные коды, код Финка 16
1.3 Кодирование числовых данных 17
1.3.1 Перевод из десятичной системы в двоичную и обратно 17
1.3.2 Перевод в восьмеричную, шестнадцатеричную системы 19
1.3.3 Дополнительный код числа 19
1.3.4 Представление вещественных чисел (мантисса, порядок) 21
1.4 Практические задания на занятие 22
2. Оформление документации в текстовых процессорах, обработка данных с использованием электронных таблиц, изучение работы в командной строке. 23
2.1 Оформление лабораторной работы в текстовых процессорах 23
2.2 Изучение электронных таблиц 28
2.3 Изучение работы в командной строке 28
3. Изучение макросов OOO Basic Libre Office Writer. 28
4. Изучение создания макросов OOO Basic Libre Office Calc. 35

Введение

Информатика это теоретическая и прикладная научная дисциплина изучающая процессы сбора, обработки, хранения и передачи информации. Мы живем в век, когда информационные процессы играют основополагающую роль в жизни человеческого общества. Трудно себе представить современную жизнь без устройств обрабатывающих, предоставляющих и хранящих в том или ином виде разнообразную информацию, требующуюся человеку как для выполнения его рабочих обязанностей, так и для улучшения его качества жизни.

Данное пособие предназначено для выполнения студентами бакалаврами практических и лабораторных работ по информатике, изучения теоретической информатики, теории кодирования и способов представления информации, а также освоения практических навыков работы с программными продуктами, использующимися для реализации информационных процессов, в частности редактирования текстов и обработки табличных данных.

1. Практическое занятие: Кодирование, представления числовой информации

1.1 Равномерное и неравномерное кодирование

Результат одного отдельного альтернативного выбора может быть представлен как 0 или 1. Тогда выбору всякого сообщения (события, символа т.п.) в массиве сообщений соответствует некоторая последовательность двоичных знаков 0 или 1, то есть двоичное слово. Это двоичное слово называют кодировкой, а множество кодировок источника источника сообщений – кодом сообщений.

Кодирование – замена информационного слова на кодовое.

Пример.

Информационное слово	Кодовое слово
000	0000
001	0011
010	0101
011	0110
100	1001
101	1010
110	1100
111	1111

Если количество символов представляет собой степень двойки ($n = 2^N$) и все знаки равновероятны $P = (1/2)^N$, то все двоичные слова имеют длину L=N=ld(n). Такие коды называют равномерными кодами.

Более оптимальным, с точки зрения объема передаваемой информации, является неравномерное кодирование, когда разным сообщениям в массиве сообщений назначают кодировку разной длины. Причем, часто происходящим событиям желательно назначать кодировку меньшей длины и наоборот, т.е. учитывать их вероятность.

Кодирование словами постоянной длины

Буква	a	b	c	d	e	f	g
Кодирование	000	001	010	011	100	101	110

ld(7)=2,807 и L=3.

1.1.1 Кодирование Шеннона Фано

Проведем кодирование, разбивая исходное множество знаков на равновероятные подмножества, то есть так, чтобы при каждом разбиении суммы вероятностей для знаков одного подмножества и для другого подмножества были одинаковы. Для этого сначала

расположим знаки в порядке уменьшения их вероятностей. Затем будем делить таблицу так, чтобы сумма вероятностей символов в одной части таблицы, была примерно равна сумме вероятностей в другой таблице, назначим одной части таблицы начальный символ 1, другой символ 0, затем продолжим делить получившиеся группы символов, так же назначая им следующие коды 1 и 0.

Итоговый результат приведен ниже, данный код получил название код Шеннона-Фано.

Символ	Вероятность, P_i	Кодировка	Длина, L_i	Вероятность*Длина, P_i*L_i
a	0,25	00	2	0,5
e	0,25	01	2	0,5
f	0,125	100	3	0,375
c	0,125	101	3	0,375
b	0,125	110	3	0,375
d	0,0625	1110	4	0,25
g	0,0625	1111	4	0,25

В общем случае алгоритм построения оптимального кода Шеннона-Фано выглядит следующим образом:

1. сообщения, входящие в ансамбль, располагаются в столбец по мере убывания вероятностей;

2. выбирается основание кода K (в нашем случае K=2); 3. все сообщения ансамбля разбиваются на K групп с

суммарными вероятностями внутри каждой группы как можно близкими друг к другу.

4. всем сообщениям первой группы в качестве первого символа присваивается 0, сообщениям второй группы – символ 1, а сообщениям K-й группы – символ (K–1); тем самым обеспечивается равная вероятность появления всех символов 0, 1,…, K на первой позиции в кодовых словах;

5. каждая из групп делится на K подгрупп с примерно равной суммарной вероятностью в каждой подгруппе. Всем сообщениям первых подгрупп в качестве второго символа присваивается 0, всем сообщениям вторых подгрупп – 1, а сообщениям K-ых подгрупп – символ (K–1).

6. процесс продолжается до тех пор, пока в каждой подгруппе не окажется по одному сообщению.

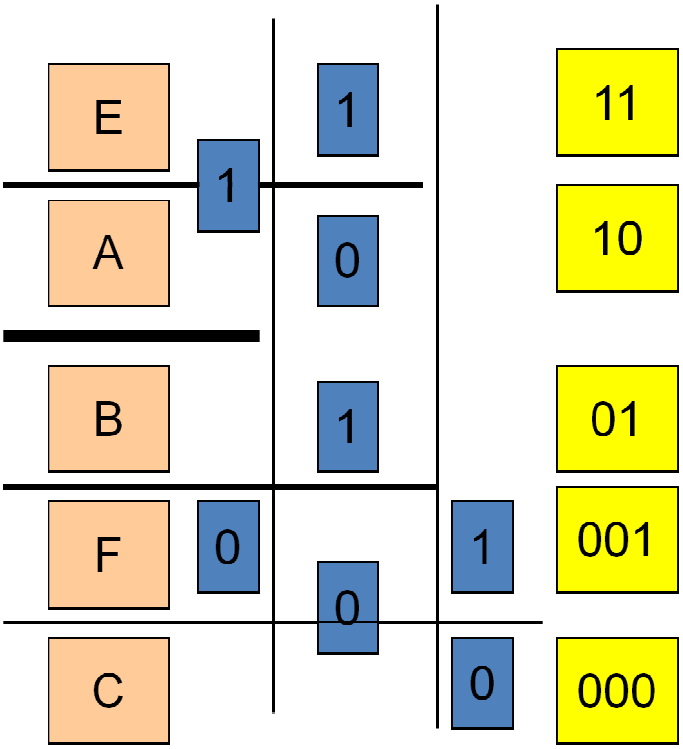
Приведем еще один пример для получения кода Шеннона-Фано. Пусть дана таблица.

Символ	Вероятность, Pi
A	0,2
E	0,4
F	0,125
C	0,125
B	0,15

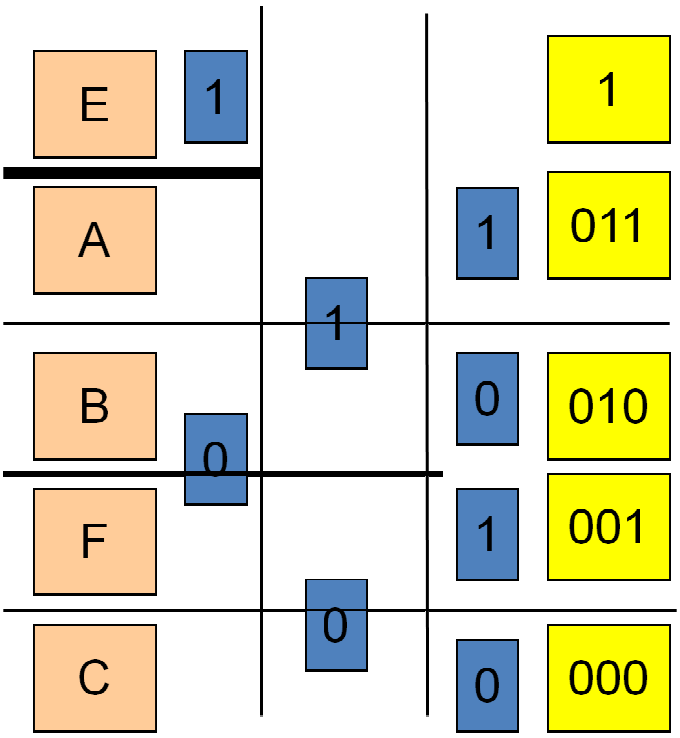
Упорядочим ее по убыванию вероятностей.

Символ	Вероятность, Pi
E	0,4
A	0,2
B	0,15
F	0,125
C	0,125

Приведем пример, где мы разделили сначала таблицу на символы групп E,A и группы (B, F, C), так как вероятности этих групп 0.6 и 0.4.



Либо разбив сначала на группы 0.4 (E), 0.6 (A, B, F, C) получим следующий результат.



Заметим, что построение кода можно представить в виде бинарного дерева, где каждой ветви назначается знак 1 или 0, например, правой 1, а левой 0.

Рассмотрим пример, где возьмем не вероятности, а частоту встречаемости символа (но это, практически, одно и то же, если поделить на общее количество встреченных символов).

A (частота встречаемости 50)

B (частота встречаемости 39)

C (частота встречаемости 18)

D (частота встречаемости 49)

E (частота встречаемости 35)

F (частота встречаемости 24)

Упорядочим:

A (частота встречаемости 50)

D (частота встречаемости 49)

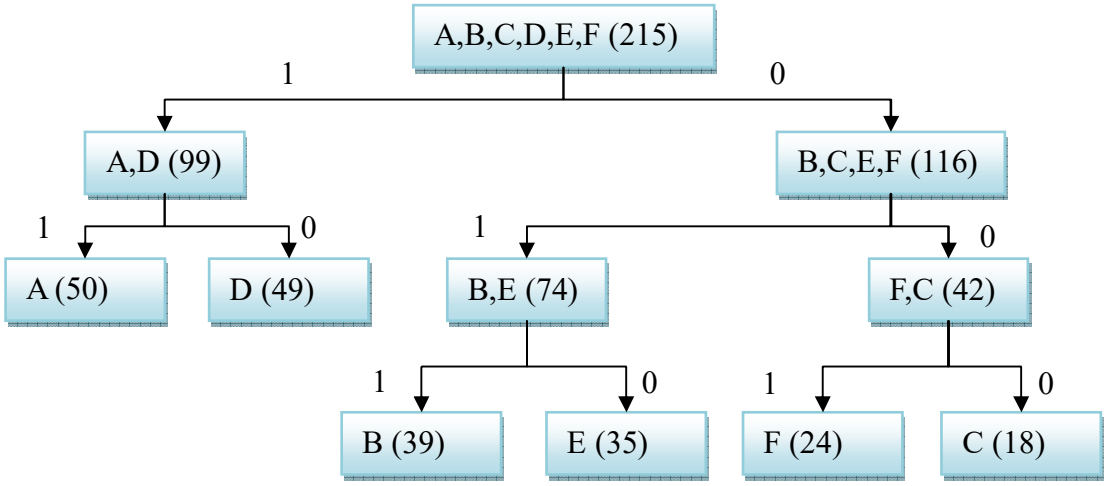
B (частота встречаемости 39)

E (частота встречаемости 35)

F (частота встречаемости 24)

C (частота встречаемости 18)

Получаем следующее дерево:



Проходим по ветвям этого дерева, записывая код для каждой буквы.

1.1.2 Кодирование Хаффмана

Еще один код, относящийся к неравномерному кодированию, это код Хафмана, он используется в процедурах сжатия без потерь.

Классический алгоритм Хаффмана на входе получает таблицу частот встречаемости символов в сообщении. Далее на основании этой таблицы строится дерево кодирования Хаффмана (Н-дерево).

1. Символы входного алфавита образуют список свободных узлов. Каждый лист имеет вес, который может быть равен либо вероятности, либо количеству вхождений символа в сжимаемое сообщение.

2. Выбираются два свободных узла дерева с наименьшими весами.

Создается их родитель с весом, равным их суммарному весу.

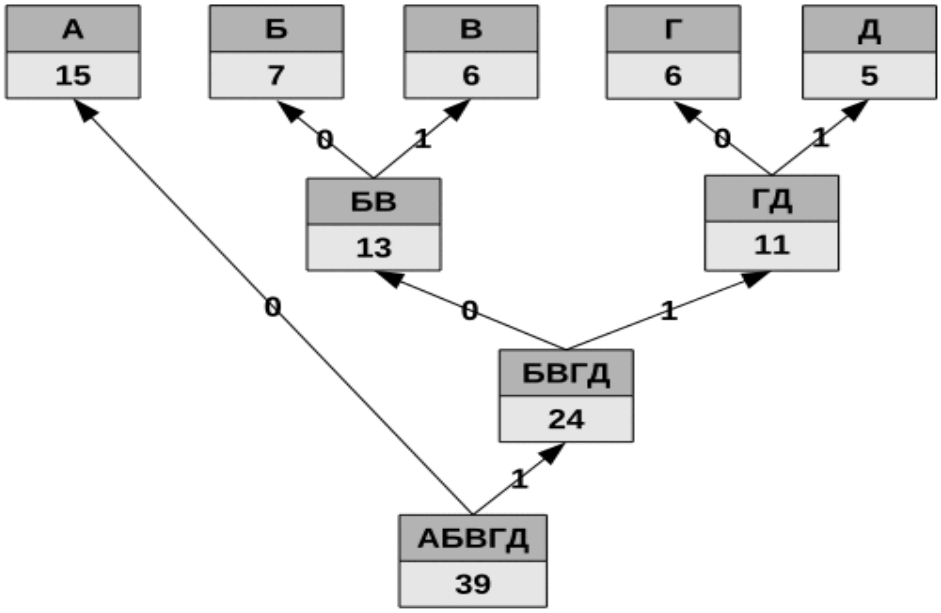
4. Родитель добавляется в список свободных узлов, а два его потомка удаляются из этого списка.

5. Одной дуге, выходящей из родителя, ставится в соответствие бит 1, другой — бит 0. Битовые значения ветвей, исходящих от корня, не зависят от весов потомков.

Шаги, начиная со второго, повторяются до тех пор, пока в списке свободных узлов не останется только один свободный узел. Он и будет считаться корнем дерева.

Проще это описать следующим образом: берем два наименее встречающиеся символа и объединяем вместе, приписывая им общую вероятность появления, одному символу при этом ставим в соответствие 1, другом ноль, теперь это один общий символ, повторяем все с этим новым символом и остальными.

Пример дерева для кодирования приведен ниже:



Итого:

А	Б	В	Г	Д
0	100	101	110	111

Здесь мы начинаем с Г и Д, далее объединяем Б и В. Потом объединяем (Г,Д) и (Б,В).

При неравномерном кодировании вводят среднюю длину кодировки, которая определяется по формуле

$$L = \sum_{i=1}^{n} P_i L_i$$

В общем же случае связь между средней длиной кодового слова L и энтропией H источника сообщений дает следующая теорема кодирования Шеннона:

имеет место неравенство L>=H, причем L = H тогда, когда набор знаков можно разбить на точно равновероятные подмножества;

всякий источник сообщений можно закодировать так, что разность L – H будет как угодно мала.

Разность L–H называют избыточностью кода (мера бесполезно совершаемых альтернативных выборов).

Символ	Вероятность	Кодировка	Длина	В×Д
A	0,7	0	1	0,7
B	0,2	10	2	0,4
C	0,1	11	2	0,2

Средняя длина слова: $L = 0,7+0,4+0,2=1,3$.

Среднее количество информации, содержащееся в знаке (энтропия):

$H = 0,7\times\text{ld}(1/0,7)+0,2\times\text{ld}(1/0,2)+0,1\times\text{ld}(1/0,1) = 0,7\times0,515+0,2\times2,322+$

$+0,1\times3,322 = =1,1571$.

Избыточность $L - H = 1,3 - 1,1571 = 0,1429$.

Следует *не просто кодировать каждый знак в отдельности*, а рассматривать вместо этого двоичные кодирования для *nk* групп по *k* знаков.

Пары	Вероятность	Кодировка	Длина	В×Д
AA	0,49	0	1	0,49
AB	0,14	100	3	0,42
BA	0,14	101	3	0,42
AC	0,07	1100	4	0,28
CA	0,07	1101	4	0,28
BB	0,04	1110	4	0,16
BC	0,02	11110	5	0,10
CB	0,02	111110	6	0,12
CC	0,01	111111	6	0,06
Средняя длина кодовой группы из 2-х символов равна				2,33

Средняя длина кода одного знака равна 2,33/2=1,165 – уже ближе к энтропии. Избыточность равна $L - H = 1,165 - 1,1571 \approx 0,008$.

Как видно при объединении в группы, избыточно стала уменьшаться и среднее число бит на символ стало ближе к энтропии.

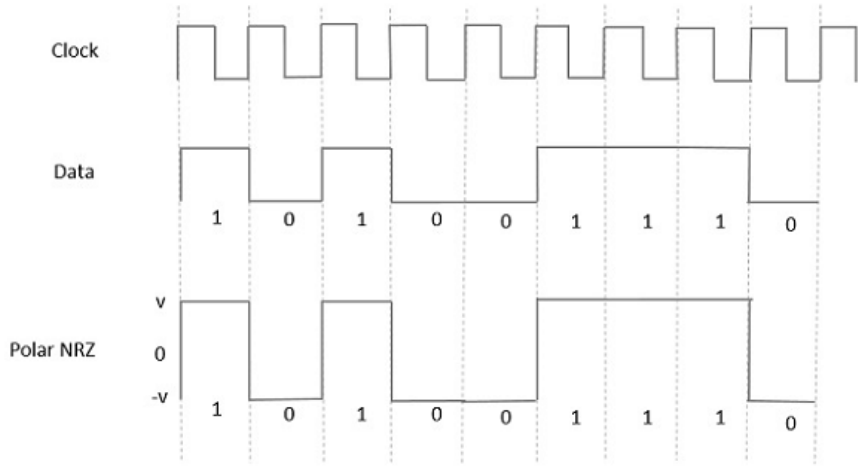
1.2 Кодирование в условиях помех

При передаче данных по каналам связи, происходит их искажение за счет наличия различных шумов, воздействующих на передающийся сигнал. Часто возникает так называемая аддитивная помеха, которая изменяет уровень сигнала, что приводит к его неправильной интерпретации в процессе демодуляции.

Используют разные способы физического кодирования, потенциальное кодирование, импульсное кодирование, выделяют методы манипуляции и модуляции, которые используют ту или иную физическую характеристику сигнала, изменяющуюся за время такта (фазу, частоту и амплитуду). Понятие манипуляция относится к цифровому сигналу, а модуляция к аналоговому.

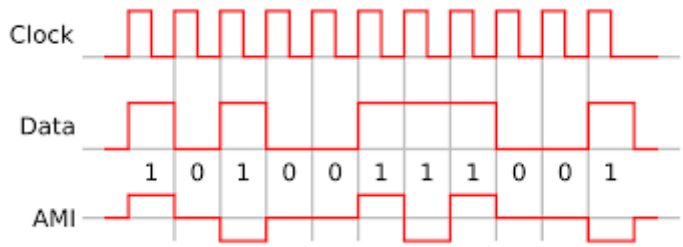
Приведем пример потенциального кода и на его примере рассмотрим основные два вида ошибок, которые возникают при передаче. Первый тип это стирание или вставка, второй тип это замена.

Потенциальный код предполагает передачу в течение такта положительного уровня для передачи 1 и нулевого уровня или отрицательного уровня для передачи 0. Рассмотрим код 101001110.



Предположим, что мы передаем данные на удаленное расстояние, тогда мы будем вынуждены прокладывать еще дополнительный тактовый канал (Clock), тратя на это провода или дополнительные ресурсы, тем более не факт что на больших расстояниях у нас не произойдет рассинхронизация. Допустим, мы попытаемся передавать информацию по одному каналу связи, используя тактовые генераторы на приемнике и передатчике, при этом пусть мы передаем много единиц или много нулей по каналу связи. Тогда на приемнике мы получаем один и тот же уровень сигнала без каких либо различий, при этом при рассинохранизации тактового генератора мы получим или лишнюю единицу, или не дополучим и пропустим ее, это и будет ошибкой вставки или стирания. Здесь мы видим, что наш код плохо самосинхронизован, потому что на приемнике не видно прихода нового символа, не происходит смена физического состояния, которое мы могли бы отличить. Потому все использующиеся физические коды самосинронизированы, например, биполярный импульсный код, код AMI, код MLT-3, манчестерский код, в них всегда есть смена состояния в такте, которая может быть различена приемником. Здесь представлен код AMI, который синхронизован по единицам, каждая новая 1 кодируется новой полярностью. Более того, так еще можно и отслеживать ошибки, если, например, за положительной полярностью следует опять положительная, а не отрицательная. Для улучшения синхронизации по нулям используются специальные коды, вводящие избыточность и запрещающие использование кодов, где, например, присутствует много

нулей подряд, например, b8zs, hdb3. Использующие коды где запрещено использование множества подряд нулей или единиц, например, 4b5b.



За счет чего происходят искажения вида замена? За счет различных физических эффектов: затухание сигнала в канале связи, поглощение или рассеяние, возможно, что эти эффекты сильнее на каких-то длинах волн, что приводит к сглаживанию сигнала, и, например, можно запутать единицу с нулем. Так же возможны помехи, которые приводят к тому, что на приемнике появляется усиленный или ослабленный сигнал, или вовсе изменивший свое состояние.

Для того, чтобы бороться с помехами типа замена используется избыточное кодирование, дополняющее информационные биты или символы дополнительными проверочными, здесь выделяют так называемое блочное кодирование и сверточное кодирование.

Прежде чем рассматривать данные методы, рассмотрим, что такое расстояние Хэмминга. Расстояние Хэмминга между двумя словами есть число разрядов, в которых эти слова различаются.

Примеры:

1. Расстояние Хэмминга d(000, 011) есть 2.
2. Расстояние Хэмминга d(10101, 11110) равно 3

В процессе декодирования происходит исправления ошибок, если они произошли. Рассмотрим пример:

Пусть есть множество кодовых слов {00000,01101,10110,11011}

Если полученное слово 10000, то декодируем в «ближайшее» слово 00000.

Если полученное слово 11000 – то только обнаружение ошибки, так как есть два варианта с минимальными до данного слова расстояниями хэмминга: 11000 – в 00000 или 11000 – в 11011.

Коды, в которых возможно автоматическое исправление ошибок, называются самокорректирующимися. Для построения самокорректирующегося кода, рассчитанного на исправление одиночных ошибок, одного контрольного разряда недостаточно.

Количество контрольных разрядов k должно быть выбрано так, чтобы удовлетворялось неравенство:

$$2^k \geq k + m + 1$$

где m количество разрядов кодируемого слова.

Минимальные значения k при заданных значениях m

Диапазон m	k_{min}
1	2
2-4	3
5-11	4
12-26	5
27-57	6

1.2.1 Блочные коды, код Хэмминга

Блочные коды помехоустойчивого кодирования основаны на добавлении к группе (блоку) информационных бит дополнительных кодовых проверочных бит, при этом размер блоков всегда постоянный и проверочные биты проверяют только данный блок, например, код Хэмминга.

Код Хэмминга является систематическим кодом, что означает, что исходные данные (сообщение) добавляются к закодированному сообщению. Биты проверки вычисляются путем выбора битов, которые являются степенями двойки (например, 1, 2, 4, 8, 16 и т.д.) и не включены в закодированное сообщение. Затем каждый бит проверки вычисляется как контрольный бит для соответствующих битов данных и битов проверки. Контрольные биты являются битами четности, которые показывают, сколько единиц есть в группе битов, для которых они являются контрольными битами. Данный код позволяет обнаружить что в коде есть 1 или две ошибки, а также может исправить одну ошибку.

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	*
2^0	2^1		2^2				2^3								2^4						
0	0	0	0	1	0	0	0	0	1	0	0	0	0	1	0	1	1	1	0	1	
X		X		X		X		X		X		X		X		X		X		X	1
	X	X			X	X			X	X			X	X			X	X			0
			X	X	X	X					X	X	X	X					X	X	1
							X	X	X	X	X	X	X	X							0
															X	X	X	X	X	X	0

Как видим из таблицы проверочные биты стоят в определенных местах соответствующих степеням двойки и обозначены желтым цветом, сначала они заполняются нулями, поля которые обозначены розовым цветом, содержат информационные биты отправляемого слова. Ниже обозначены в строках разными цветами и X проверочные

подпоследовательности, например, первый проверочный бит будет проверяться первой последовательностью на четность, достаточно сложить по модулю два или операцией xor все биты в отмеченных позициях. Запишем все вычисления для каждой последовательности, здесь плюс это операция xor (напомним таблицу истинности, 0+0=0, 1+0=1,0+1=1, 1+1=0):

0+0+1+0+0+0+0+1+1+1+1=1

0+0+0+0+1+0+0+1+1+1=0

0+1+0+0+0+0+0+1+0+1=1

0+0+1+0+0+0+0+1=0

0+1+1+1+0+1=0.

Можно так же получить остаток от деления на два после суммирования (потому и называется сложение по модулю два), проверить на четность-нечетность.

Теперь вставим указанные биты в соответствующие позиции по порядку:

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	
2^0	2^1		2^2				2^3								2^4						
1	0	0	1	1	0	0	0	0	1	0	0	0	0	1	0	1	1	1	0	1	

Теперь можно передавать это сообщение по сети и в случае одиночной ошибки исправить ее. Проверим это утверждение, сделав намеренное искажение одиннадцатого бита.

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	*
2^0	2^1		2^2				2^3								2^4						
0	0	0	0	1	0	0	0	0	1	1	0	0	0	1	0	1	1	1	0	1	
X		X		X		X		X		X		X		X		X		X		X	1
	X	X			X	X			X	X			X	X			X	X			1
			X	X	X	X					X	X	X	X					X	X	0
							X	X	X	X	X	X	X	X							1
															X	X	X	X	X	X	0

Проведем те же операции что и раньше:

1+0+1+0+0+1+0+1+1+1+1 = 1 1

0+0+0+0+1+**1**+0+1+1+1 = 1 2

1+1+0+0+0+0+0+1+0+1 = 0 4

0+0+1+**1**+0+0+0+1 = 1 8

0+1+1+1+0+1 = 0 16

Если теперь сложить полученные коды с соответствующими весами степени двойки, мы

получим число 11, которое соответствует номеру ошибочного бита, можно теперь его исправить.

$01011_2=11_{10}$

Можно интерпретировать еще код Хэмминга следующим образом: смотрим, например, последовательность, если в ней ошибка есть значит ошибка среди бит данной последовательности, если ошибки в этой последовательности нет, значит тут правильные биты и можно методом исключения выбрать неправильный бит. В нашем случае, например, ошибка в 1, 2 и 4 последовательностях, значит ошибка в 11 или 15 бите, но в третьей последовательности ошибки нет, значит ошибка в 11.

Почему в методе Хэмминга удается получить номер искаженного бита? В том числе благодаря свойству xor и тому что каждый бит под своим номером проверяется последовательностью, которая имеет определенный вес для кодирования данного числа, и когда искажается именно эти последовательности, мы получаем номер искаженного бита. Например, первый бит проверяется только первой последовательностью, потому ее искажение выделяет нам первый бит – 1, при искажении второго бита исказится и первая и вторая последовательности.

1.2.2 Сверточные коды, код Финка

Сверточные коды получаются путем вставки между последовательностями информационных бит проверочных, при этом проверка потоковая, нет деления на блоки. Витерби показал, что наиболее оптимальными являются сверточные коды.

Например, рекуррентный код Финка получается следующим образом:

- $A_1,B_1, A_2, B_2, A_3, B_3... A_k, B_k A_{k+1}, B_{k+1}....$
- $B_k=A_{i-s}\oplus A_{i+s+1}$ расчет проверочного символа B.

С учетом данного способа, ошибка исправляется если в проверочных битах k1=i-s-1, и k2=i+s ошибка, и k2-k1=2s+1. Например, s=0, B1 и B2 неверные значит A2 инвертировать. Сам проверочный символ k=i неверен если ошибка при A_{i-s}, A_{i+s+1}

Часто сверточные коды описываются многочленами.

Допустим, зададим информационную последовательность:

$A(x)=a_0+a_1X+a_2X^2+\dots$ a {0,1..}

Формируемые передаваемые биты могут описываться следующим образом:

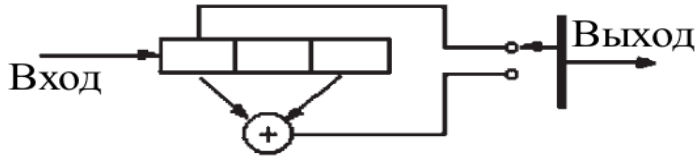
$B_j(X)=b_0+b_1X+b_2X^2+\dots$

$B_j(X)=G_j(X)A(X)$

где G – это порождающие полиномы.

При этом систематические коды не меняют информационную последовательность, а несистематические меняют.

Пример сверточного кода Финка в терминах порождающих полиномов с s=1:

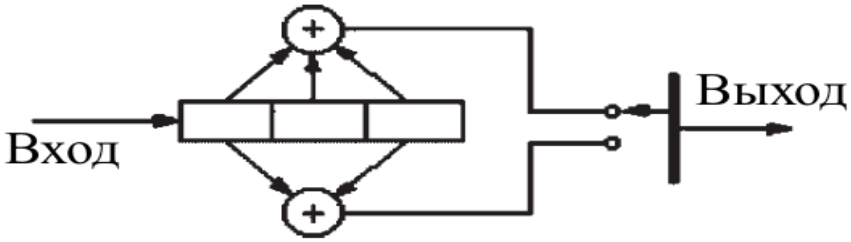


$G_1(X)=1$; $G_2(X)=1+X^2$

Для кода Финка с s=0 порождающие полиномы имеют вид:

$G_1(X)=1$; $G_2(X)=1+X$;

Приведем еще пример несистематического сверточного кода:



$G_1(X)=1+X+X^2$; $G_2(X)=1+X^2$

1.3 Кодирование числовых данных

Система счисления это способ записи числа с помощью специального набора знаков.

Известно, что число с дробной частью в позиционной системе счисления можно представить следующим образом:

$$p = \sum_{i=-m}^{n} a_i p^i$$

где a_i – цифра числа в позиции, p – основание системы счисления.

Таким образом, позиция цифры определяет вес цифры в числе, определяемый основанием системы счисления в степени номера позиции. Отрицательные номера определяют дробную часть числа, положительные и нулевая степень целую часть числа.

1.3.1 Перевод из десятичной системы в двоичную и обратно

Один из способов перевода числа из какой-либо системы в десятичную заключается в разложение на соответствующие степени и выполнения операций в десятичной системе для получения числа в десятичной системе, так же можно перевести из десятичной системы в любую другую, реализовав разложение, начиная с большей степени, так чтобы в итоге получить исходной число.

Например, переведем из десятеричной системы счисления число 25_{10} в двоичную

систему счисления. Первая степень, которая входит в данное число это 16, остается 9, потом входит 8, и затем 1. Степени, которые входят, помечаем как 1, те которые не входят в число, помечаем как 0.

2^4	2^3	2^2	2^1	2^0
16	8	4	2	1
1	1	0	0	1

Можем проверить, сложив соответствующие степени, где стоит 1 и получим 25.

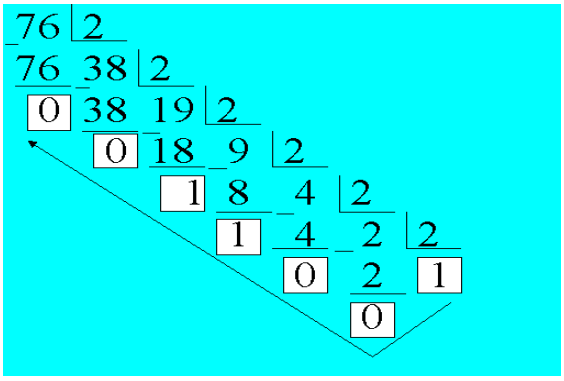
Обратно из двоичной перевести так же просто, допустим число 10111_2.

$1*2^4+0*2^3+1*2^2+1*2^1+1*2^0=23_{10.}$

$111000.11_2=1•2^5+1•2^4+1•2^3+1•2^{-1}+1•2^{-2} = 32 + 16 + 8 + ½ + ¼ = 56,75_{10}$

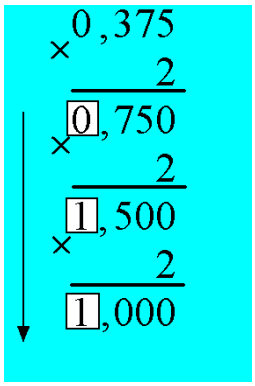
Так же можно переводить из одной системы счисления в десятичную операцией деления, делим на основание системы счисления, получаем остаток от деления и записываем его в качестве младшей цифры числа, затем результат от целого деления продолжаем делить на основание системы счисления, пока не получим 0.

$76_{10} = 1001100_{2.}$



Очевидно, здесь для последней 1 можно не брать остаток от деления на 2, итак ясно что нужно записать 1 как значение в старшем разряде.

Дробная часть получается из целых частей (0 или 1) при ее последовательном умножении на 2 до тех пор, пока дробная часть не обратится в 0 или получится требуемое количество знаков после разделительной точки.



$0{,}375_{10}=0.011_2$

1.3.2 Перевод в восьмеричную, шестнадцатеричную системы

Здесь можно использовать основание системы счисления:

$421.5_8 = 4\cdot8^2+2\cdot8^1+1\cdot8^0+5\cdot8^{-1} = 256 + 16 + 1 + 5/8 = 273{,}625_{10}$

Пример для шестнадцатеричной системы счисления:

$A7.C_{16} = 10\cdot16^1+7\cdot16^0+12\cdot16^{-1} = 160 + 7 + 12/16 = 167{,}75_{10}$

Либо если перевести в двоичную систему и далее разбивать представления на тройки или четверки бит начиная с младшего разряда для целой части и со старшего разряда для дробной части..

$100110111.001_2 = 467.1_8$

$100110111.001_2 = 137.2_{16}$

$101001011110.11_2 = 52E.C_{10}$

Проще всего в данном случае чтобы запомнить соответствие между кодами и буквами в шестнадцатеричной системе счисления, перевести двоичное число в десятичное, например, 1010 = 10, тогда после девяти идет А, тогда заменяем на А. Для некратного количества цифр можно для целых дополнить нулями слева и для дробных справа.

1.3.3 Дополнительный код числа

Дополнительный код - это способ представления отрицательных чисел в двоичной системе счисления. Он используется для упрощения арифметических операций с отрицательными числами. В дополнительном коде отрицательные числа представляются как инвертированный двоичный код положительного числа, увеличенного на единицу. Например, число -5 в дополнительном коде будет представлено как 11111011. При выполнении операций с числами в дополнительном коде, результат также будет представлен в дополнительном коде. Дополнительный код используется во многих компьютерных системах, включая процессоры и операционные системы. Можно заменить операцию вычитания на операцию сложения с числом, представленным в дополнительном коде.

Также число в дополнительном коде получается как вычитание из самого минимального числа со значащей цифрой в большей на единицу разрядной сетке, чем исходное число. Например, в десятичной системе счисления для числа 67, будет 100-67 = 33. Для числа 259, 1000 - 259 = 741. Можно легко заметить что, число с минимальной цифрой 1 в большей на единицу разрядной сетке, можно представить как максимальное число в данной разрядной сетке плюс 1. Например 1000 = 999+1. При этом, запишем:

1000-x = (999-xxx)+1. Операция 999-xxx может быть представлена как инвертирование, получаемое соответствием 9-0, 8-1, 7-2, 6-3, 5-4, 4-5, 3-6, 2-7, 1-8, 0-9.

Допустим, мы проводим вычетание:

xxx-yyy, представим что для yyy мы нашли дополнительный код, который равен zzz = 1000-yyy, очевидно при этом yyy = 1000-zzz, тогда:

xxx – (1000-zzz) = xxx+zzz-1000, таким образом мы получаем сложение с дополнительным кодом, вместо вычитания.

При этом действительно очевидным становится, что для получения дополнительного кода в двоичной системе счисления достаточно провести и инвертирование числа и сложить с единицей, например, инвертирование делается просто элементом «Не», и далее делается инкрементация с помощью регистра на Т-триггерах.

Найти разность $8_{10} - 13_{10}$ в восьмибитном представлении.

	8_{10}	-13_{10}
Прямой код	**00001000**	**10001101**
Обратный код	**-**	**11110010**
Дополнительный код	**-**	**11110011**

0 0 0 0 1 0 0 0

1 1 1 1 0 0 1 1

1 1 1 1 1 0 1 1

Теперь возьмем дополнительный код от полученного числа:

Инвертируем

0 0000 1 00

Прибавим 1

0 0000 1 00

0 0000 0 01

0 0000 1 01 = 5. Все правильно.

Кроме того, для получения отрицательного числа в дополнительном коде с большим числом разрядов, данное число расширяется единицами. Например, при расширении от байта к слову, char к short.

1.3.4 Представление вещественных чисел (мантисса, порядок)

Вещественные числа хранятся и обрабатываются в компьютере в формате *с плавающей запятой*, использующем экспоненциальную форму записи чисел.

$$A = M \times q^n$$

где M – мантисса числа (правильная отличная от нуля дробь), q – основание системы счисления, n – порядок числа.

Диапазон ограничен максимальными значениями M и n.

Например, $123{,}45 = 0{,}12345 \cdot 10^3$

Порядок указывает, на какое количество позиций и в каком направлении должна сместиться десятичная запятая в мантиссе.

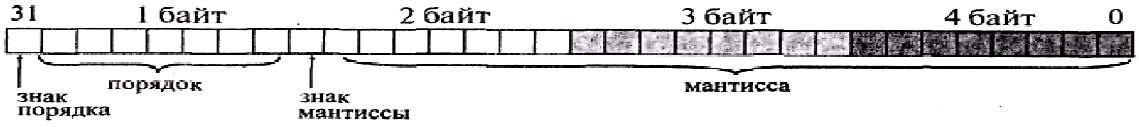
Число в формате с плавающей запятой может занимать в памяти 4 байта *(обычная точность)* или 8 байтов *(двойная точность)*.

При записи числа выделяются разряды для хранения знака мантиссы, знака порядка, порядка и мантиссы.

Мантисса **M** и порядок **n** определяют диапазон изменения чисел и их точность.

Число в форме с плавающей точкой занимает в памяти компьютера **четыре** (число обычной точности) байта или **восемь** (число двойной точности) байта.

Для записи чисел в разрядной сетке выделяются разряды для знака порядка и мантиссы, для порядка и для мантиссы.



Пример.

Представить число 250,187510 в формате с плавающей точкой в 4-байтовой разрядной сетке.

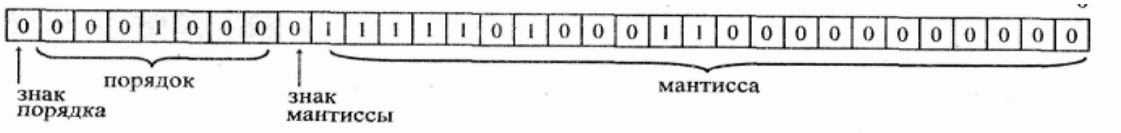
Переведем число в двоичную систему счисления с 23 значащими цифрами:

$250{,}1875_{10} = 11111010{,}001100000000000_2$;

Нормализуем мантиссу: 11111010,001100000000000 = 0,111110100011 00000000000·10^{1000};

0,11111010001100000000000 · 10^{1000}; (мантисса положительная), (порядок положительный).

Запишем число в 32-разрядной сетке:



1.4 Практические задания на занятие

Пусть заданы числа:

d – день рождения, если d<10, то 01, 02 и т.д.

m – месяц, если m<10, то 01, 02 и т.д.

Получим число x=d+m+40 в десятичной системе.

1) x перевести в двоичную, 8-ю, 16-ю системы счисления.

2) 1100110011.10101_2 – перевести в десятичную

Пусть

k = m/3+(16-m)/4 – округлить до целых

k(единичек)001.100k(единичек), например k=2, 11001.10011 – перевести в десятичную, восьмеричную, шестнадцатеричную.

Пусть даны числа:

$x.d_{10}$, $d.x_{10}$, $-0.000xd_{10}$, $xd445_{10}$

Перевести в двоичную систему счисления, перевести в представлении мантисса-порядок (учесть знак порядка и мантиссы, отвести 2, 4 байта под число).

Для данных выражений:

-x+50, -50+x, -50-x – вычислить используя дополнительный код.

Даны символы a,b,c,d,e,f,g, вероятность a – 0.d, вероятность b – 0.m, c – 0.2, d – 0.1, e – 0.1, f – 0.05, g - 1-0.m-0.d-0.45, закодировать методом Шеннона-Фано и Хафмана, равномерным кодом, посчитать среднее число бит на символ, посчитать энтропию источника сообщения, сравнить равномерный код и неравномерный.

X_2 – закодировать кодом Хэмминга и проверить ошибку искажения одного бита. Привести пример сверточного кода Финка.

Посчитать пропускную способность канала связи если, полоса пропускания равна 100 КГц, уровень сигнала 5 Вт, уровень шума 1 Вт.

Посчитать скорость передачи оцифрованного звукового сигнала с частотой 8000 Гц и 8, 16, 256 уровнями квантования.

Контрольные вопросы:

Какие методы сжатия без потерь и с потерями вы знаете.

Данные, знания, свойства знаний, энтропия, информация в узком смысле

Лазерный принтер.

CD-Rom.

Струйный принтер.

Наборно ассоциативный кэш.

Оптическая мышка.

Флэш память.

Матричный принтер.

Ассоциативный кэш.

Механическая мышка

Кэш прямого отображения.

Монитор на ЭЛТ.

Набор регистров и основные характеристики процессора 8086.

ЖК-Монитор.

Прерывания

Плазменный монитор.

Супер-скалярный процессор.

Сканнер.

Конвейерное исполнение команд

Жесткий диск.

Машина фон-неймана. Гарвардская и Принстонская архитектуры.

2. Оформление документации в текстовых процессорах, обработка данных с использованием электронных таблиц, изучение работы в командной строке.

2.1 Оформление лабораторной работы в текстовых процессорах

Ознакомить студентов с требованиями к оформлению лабораторных работ. Дать основные навыки по оформлению автоматического содержания, расстановки номеров страниц, альбомного листа внутри текста с книжными страницами, дать навыки выделения и копирования текста, оформления формул, таблиц, рисунков, использованных источников, поиска и замены текста и специальных символов, изменения стиля текста, абзаца и страницы.

Лабораторная работа – небольшой научный отчет, обобщающий проведенную студентом работу, которую представляют для защиты преподавателю. К лабораторным работам предъявляется ряд требований, основным из которых является полное,

исчерпывающее описание всей проделанной работы, позволяющее судить о полученных результатах, степени выполнения заданий и профессиональной подготовке студентов.

В отчет по лабораторной работе должны быть включены следующие пункты:

- титульный лист;
- введение с указанием цели работы;
- краткие теоретические сведения;
- описание хода работы и полученные результаты;
- выводы.

Требования к содержанию отдельных частей отчета по лабораторной работе

Титульный лист является первой страницей любой научной работы и для конкретного вида работы заполняется по определенным правилам. Для лабораторной работы титульный лист оформляется следующим образом.

В верхнем поле листа указывают полное наименование учебного заведения и кафедры, на которой выполнялась данная работа.

В среднем поле указывается вид работы, в данном случае лабораторная работа с указанием курса, по которому она выполнена, и ниже ее название. Название лабораторной работы приводится без слова тема и в кавычки не заключается.

Далее ближе к правому краю титульного листа указывают фамилию, инициалы, курс и группу учащегося, выполнившего работу, а также фамилию, инициалы, ученую степень и должность преподавателя, принявшего работу.

В нижнем поле листа указывается место выполнения работы и год ее написания (без слова год).

Образец написания титульного листа лабораторной работы приведен ниже.

Введение и цель работы должны отражать тему лабораторной работы, а также конкретные задачи, поставленные студенту на период выполнения работы. По объему цель работы в зависимости от сложности и многозадачности работы составляет от нескольких строк до 0,5 страницы.

Краткие теоретические сведения. В этом разделе излагается краткое теоретическое описание изучаемого в работе предмета. Объем литературного обзора не должен превышать 1/3 части всего отчета.

Результаты и ход лабораторный работы. В этом разделе приводятся непосредственно результаты, полученные в ходе проведения лабораторных работ: графики, таблицы, диаграммы, последовательность выполняемых действий.

Выводы. В выводах кратко излагаются результаты работы.

Отчет по лабораторной работе оформляется на писчей бумаге стандартного

формата А4 на одной стороне листа, которые сшиваются в скоросшивателе или переплетаются. Допускается оформление отчета по лабораторной работе в электронном виде средствами Microsoft Office Word или OpenOffice Write, либо LibreOffice Writer.

Используется формат А4 через 1,5 интервала с числом строк на странице не более 40. Текст работы следует писать, соблюдая следующие размеры полей:

левое – не менее 30 мм;

правое – не менее 10 мм;

верхнее – не менее 15 мм;

нижнее – не менее 20 мм.

Размер шрифта не менее 12 пт.

Тест работы делится на главы, разделы, подразделы, пункты.

Оформление таблиц

Таблицы помещают непосредственно после абзацев, содержащих ссылку на них, а если места недостаточно, то в начале следующей страницы.

Таблицы снабжают тематическими заголовками, которые располагаются посередине страницы и пишут прописным шрифтом без точки на конце. Напр.:

Таблица 1. – Список студентов

ФИО	Год рождения	Год поступления
Иванов И.И.	1990	2010
Петров А.А.	1991	2011

При переносе таблицы на следующую страницу головку таблицы следует повторить и над ней поместить Продолжение табл. 1”. При переносе таблицы на другую страницу нумеруют заголовки граф. Тогда на новой странице заголовки граф заменяют цифрами. Тематический заголовок при этом можно не повторять.

Математические формулы

Формулы отделяются от последующего и предыдущего текста (или других формул) одной строкой.

Несколько коротких однотипных формул можно помещать в одной строке, а не в столбик.

Нумеровать следует только наиболее важные формулы, на которые в тексте имеются ссылки.

Порядковые номера ставят в круглых скобках у правого края листа. Нумерация может быть сквозной или поглавной.

Например:

$$H = -\sum_i p_i \log_2(p_i) \qquad (1.3.2)$$

где, H - энтропия, p_i - вероятность i - го символа в сообщении.

Иллюстративный материал

В качестве иллюстраций можно использовать фотографии, рисунки, чертежи, схемы, диаграммы, номограммы. Размеры иллюстраций не должны превышать формата страницы с учетом полей.

В тексте, где идет речь о теме, связанной с иллюстрацией, помещают ссылку либо в виде заключенного в круглые скобки выражения (рисунок 3) либо в виде оборота типа …как это видно на рисунке 3. Ниже приведен пример.

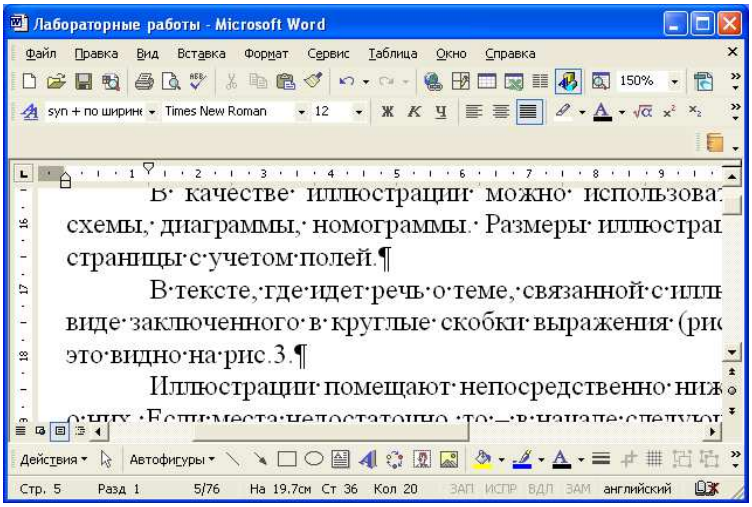


Рисунок 3 – Рабочее окно текстового процессора Microsoft Word

Иллюстрации помещают непосредственно ниже абзацев, содержащих упоминание о них. Если места недостаточно, то – в начале следующей страницы.

Нумерация рисунков может быть как сквозной, так и индексационной поглавной.

Если рисунок в книге (статье) один, то он не нумеруется. Пояснение частей иллюстрации, расшифровку условных обозначений можно включить в состав подписи.

Библиографическое описание делается в зависимости от ГОСТа, они бывают разных годов, потому лучше ориентироваться на официальные документы организации.

В качестве приложений используют дополнительный материал, чаще всего вспомогательного характера: образцы выполнения работ, расчетов, разного рода таблицы, формы, таблицы, схемы и т.п.

Приложения располагают в конце издания после списка литературы. Слово Приложение пишут справа вверху. Если приложений несколько, то их нумеруют. Знак № и точку не ставят. Можно выделить разрядкой, курсивом или прописными буквами.

Далее приводится пример титульного листа.

Министерство науки и высшего образования
Российской Федерации
Федеральное государственное бюджетное образовательное учреждение высшего образования
ТОМСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
СИСТЕМ УПРАВЛЕНИЯ И РАДИОЭЛЕКТРОНИКИ (ТУСУР)

КАФЕДРА АВТОМАТИЗИРОВАННЫХ СИСТЕМ УПРАВЛЕНИЯ

Лабораторная работа 1

по курсу «Информатика»

Название лабораторной работы

Выполнил:
Студент 1-го курса
гр. номер
_____Фамилия И.О.
подпись

_____
дата

Принял:
к.т.н., доцент
_____ _____Фамилия И.О.
Оценка подпись

_____
дата

Томск 2020

2.2 Изучение электронных таблиц

Изучить основные возможности электронных таблиц по обработке данных, включая расчеты по данным в ячейках, оформление таблиц, использование закрепления ячеек, условного оператора в расчетах, построения графиков, использование агрегированных функций и расчетов по условию, сводных таблиц, итоговых таблиц и фильтрации.

2.3 Изучение работы в командной строке

Освоить работу в командной строке в одной из операционных систем, в частности, операционных систем семейства Windows (cmd).

Копирование файлов, создание файлов, просмотр содержимого каталогов, удаление каталогов и файлов, выбор логических дисков, задание маски файлов, перенаправление вывода результатов команд в файл, использование относительного пути. Пример написания командного файла.

3. Изучение макросов OOO Basic Libre Office Writer.

Изучить основные возможности работы с макросами и документами. Для поиска необходимых функций также можно воспользоваться помощником Libre Office и официальным сайтом.

Для получения доступа к свойствам, объектам и методам документа Writer, необходимо обратиться к ссылке StarDesktop и его свойству CurrentComponent.

1) Написать макросы для очистки формата всего текста документа, первого абзаца.

Подсказка. Перебрать все параграфы и задать одни и те же общие свойства текста параграфа.

```
Sub Main
Dim Doc As Object
Dim Cursor As Object
Dim Proceed As Boolean
Doc = StarDesktop.CurrentComponent
Enum1 = Doc.Text.createEnumeration
' цикл по всем абзацам
While Enum1.hasMoreElements
TextElement = Enum1.nextElement
'проверка является ли текстовый элемент параграфом
```

```
If TextElement.supportsService("com.sun.star.text.Paragraph") Then
TextElement.CharHeight = 20
'здесь добавить остальные изменяющиеся свойства текста для их изменения.
End If
Wend
End Sub
```

Ниже приведен список свойств текста.

• CharFontName (String) – имя выбранного типа шрифта;

Например, TextElement.CharFontName = "Free Times".

• CharColor (Long) – цвет текста;

Например, TextElement.CharColor = RGB(0,255,0) — зеленый цвет

• CharHeight (Float) – высота символа в пунктах (pt);

• CharUnderline (Constant group) – тип подчеркивания (константы в соответствии с com.sun.star.awt.FontUnderline);

Например, TextElement.CharUnderline = com.sun.star.awt.FontUnderline.WAVE

• CharWeight (Constant group) – вес шрифта (константы в соответствии с com.sun.star.awt.FontWeight);

• CharBackColor (Long) – фоновый цвет;

• CharKeepTogether (Boolean) – подавление автоматического разрыва строк;

• CharStyleName (String) – имя стиля символа.

Курсив устанавливается в свойстве шрифта CharPosture, присвоив данной переменной значения 2 TextElement.CharPosture = 2 или TextElement.CharPosture = com.sun.star.awt.FontSlant.ITALIC. Кроме italic в FontSlant есть свойства NONE, OBLIQUE, ITALIC DONTKNOW, REVERSE_OBLIQUE, REVERSE_ITALIC.

Также свойства абзаца.

• ParaAdjust (enum) – вертикальная ориентация текста (константы в соответствии с com.sun.star.style.ParagraphAdjust);

• ParaLineSpacing (struct) – межстрочный интервал (структура в соответствии с com.sun.star.style.LineSpacing); Константы LineSpacingMode PROP - высота пропорциональна, MINIMUM — минимальная высота строки, LEADING — расстояние до предыдущей линии, FIX — фиксированная высота строки.

Например,

```
v = TextElement.ParaLineSpacing
```

```
v.Mode = com.sun.star.style.LineSpacingMode.FIX
v.Height = 300
TextElement.ParaLineSpacing =  v
```

• ParaBackColor (Long) – фоновый цвет;

• ParaLeftMargin (Long) – левое поле в сотых долях миллиметра;

• ParaRightMargin (Long) – правое поле в сотых долях миллиметра;

• ParaTopMargin (Long) – верхнее поле в сотых долях миллиметра;

• ParaBottomMargin (Long) – нижнее поле в сотых долях миллиметра;

• ParaTabStops (Array of struct) – тип и положение позиций табуляции (массив структур типа com.sun.star.style.TabStop);

• ParaStyleName (String) – имя стиля абзаца.

2) Написать макрос для изменения стиля каждой первой и пятой буквы каждого абзаца на курсив, изменить цвет буквы в активном документе и выделенном тексте.

Ниже приведен макрос, который используется для того, чтобы каждое первое слово предложения было выделено жирным шрифтом. При этом создается специальный объект текстовый Cursor, который позволяет осуществлять навигацию по документу, выделяя нужный текст или область нужного текста. Мы можем устанавливать все свойства символов для данного выделенного текста, при этом нужно помнить, что текст не является выделенным в понимании - выделение для копирования, а просто это область текста или блок текста свойства, которого будут меняться если мы будем присваивать какие-то значения свойствам объекта Cursor.

```
Sub Main
Dim Doc As Object
Dim Cursor As Object
Dim Proceed As Boolean
Doc = StarDesktop.CurrentComponent
'создать объект — текстовый курсор для текущего открытого документа
Cursor = Doc.Text.createTextCursor()
' начать цикл
Do
' перейти к концу слова с выделением от текущего положения курсора
Cursor.gotoEndOfWord(True)
'изменить шрифт отмеченного текста на жирный
```

```
Cursor.CharWeight = com.sun.star.awt.FontWeight.BOLD
'перейти на следующее предложение без выделения, в переменную proceed занести значение возвращаемой функцией  gotoNextSentence, она вернет True если следующее предложение есть и False если нет, т.е. фактически когда завершится текст функцяи вернет значение False
Proceed = Cursor.gotoNextSentence(False)
'перейти на начало текущего предложения без выделения
Cursor.gotoStartOfSentence(False)
'условие выполнения цикла, пока логическая переменная Proceed истинна, то-есть True
Loop While Proceed
msgbox "end"
End Sub
```

Ниже перечислены возможные способы навигации.

- goLeft (Count, Expand) – переход на Count символов влево;
- goRight (Count, Expand) – переход на Count символов вправо;
- gotoStart (Expand) – переход к началу текстового документа;
- gotoEnd (Expand) – переход к концу текстового документа;
- gotoRange (TextRange, Expand) – переход к указанному TextRange-объекту;
- gotoStartOfWord (Expand) – переход к началу текущего слова;
- gotoEndOfWord (Expand) – переход к концу текущего слова;
- gotoNextWord (Expand) – переход к началу следующего слова;
- gotoPreviousWord (Expand) – переход к началу предыдущего слова;
- isStartOfWord () – возвращает True, если TextCursor в начале слова;
- isEndOfWord () – возвращает True, если TextCursor в конце слова;
- gotoStartOfSentence (Expand) – переход к началу текущего предложения;
- gotoEndOfSentence (Expand) – переход к концу текущего предложения;
- gotoNextSentence (Expand) – переход к началу следующего предложения;
- gotoPreviousSentence (Expand) – переход к началу предыдущего предложения;
- isStartOfSentence () – возвращает True, если TextCursor в начале предложения;
- isEndOfSentence () – возвращает True, если TextCursor в конце предложения;
- gotoStartOfParagraph (Expand) – переход к началу текущего абзаца;
- gotoEndOfParagraph (Expand) – переход к концу текущего абзаца;

• gotoNextParagraph (Expand) – переход к началу следующего абзаца;

• gotoPreviousParagraph (Expand) – переход к началу предыдущего абзаца;

• isStartOfParagraph () – возвращает True, если TextCursor в начале абзаца;

• isEndOfParagraph () – возвращает True, если TextCursor в конце абзаца.

Входной параметр Expand показывает выделяется ли текст при передвижении курсора, значение True — выделяется (отмечается) и False — курсор продвигается и текст не выделяется (не отмечается). Каждая функция при этом возвращает значение true, либо false в зависимости от успешности выполнения, например, если следующего параграфа нет при вызове функции перейти на следующий параграф, то вернется значение false, если далее параграф есть, то возвращается True.

Подсказка для выполнения задания:

Использовать gotoStartOfParagraph, .goRight, CharColor , gotoNextParagraph.

3) Написать макрос для поиска в тексте запятых и замены их на троеточие.

Ниже приведен пример макроса, который позволяет заменить одни слова на другие, используя свойство параграфа textportion, это текст являющийся частью параграфа, которому присущи свои собственные свойства и характеристики.

```
Sub Main
Dim Doc As Object
Dim Enum1 As Object
Dim Enum2 As Object
Dim TextElement As Object
Dim TextPortion As Object
Doc = StarDesktop.CurrentComponent
Enum1 = Doc.Text.createEnumeration
' цикл по всем абзацам
While Enum1.hasMoreElements
TextElement = Enum1.nextElement
'проверка является ли текстовый элемент параграфом
If TextElement.supportsService("com.sun.star.text.Paragraph") Then
Enum2 = TextElement.createEnumeration
' цикл по всем элементам текущего параграфа (текстовым порциям) TextElement, пока есть еще элементы
```

```
While Enum2.hasMoreElements
TextPortion = Enum2.nextElement
```

' взять строку текста порции текста, произвести замену одной последовательности символов на другую, эту замену возвратит функция replace, произвести присвоение строке содержащейся в порции новой строки возвращенной функцией replace.

```
TextPortion.String = Replace(TextPortion.String, "you", "U")
TextPortion.String = Replace(TextPortion.String, "o", "2")
Wend
End If
Wend
msgbox s
End Sub
```

4) Сделать макрос для обмена двух абзацев местами.

Можно воспользоваться свойством параграфа TextElement.String и поменять текст в двух параграфах местами. Заведите две переменные TextElement, присвойте им значение первого и последующего параграфа с помощью Enum.nextElement и используя третью строковую переменную произведите обмен.

5) Изменить цвет и размер шрифта каждого абзаца на произвольный. Использовать функцию rnd(), которая возвращает случайное вещественное значение от 0 до 1 и функцию rgb(r,g,b), которая возвращает преобразованное цветовое значение из последовательности интенсивностей красного, зеленого и синего, где каждое значение интенсивности варьируется от 0 до 255. Комбинация интенсивностей основных трех цветов создает для человека иллюзию различных цветов, например, три интенсивности со значениями 255, будет давать белый цвет, все значения 140, дают серый цвет.

6) Изменить цвет и размер шрифта каждого третьего слова в тексте.

Ниже приведен пример макроса, который меняет цвет каждой первой буквы параграфа на красный цвет.

```
Sub Main
Dim Doc As Object
Dim Cursor As Object
Dim Proceed As Boolean
Doc = StarDesktop.CurrentComponent
Cursor = Doc.Text.createTextCursor()
Do
```

```
Cursor.gotoStartOfParagraph(False)
Cursor.goRight(1,True)
Cursor.CharColor =  RGB(255,0,0)
Proceed = Cursor.gotoNextParagraph(False)
Loop While Proceed
msgbox "end"
End Sub
```

7) Написать макрос, в котором каждая запятая будет заменена на последовательность трех разноцветных точек. Подсказка: использовать объект Cursor и его свойство String, данное свойство содержит текущую выделенную строку, если использовались функции движения по тексту со значением True, можно выделить одну букву, сравнить ее с запятой и заменить на три точки. Не забывайте сбрасывать выделение с помощью движения со значением False.

8) Выполнить индивидуальный макрос по вариантам ниже.

Задания. Макросы Writer.

1. В каждом втором слове каждого второго параграфа заменить первую половину слова на вторую. Цвет для первой половины задать инвертированным по красной составляющей второго слова. Все буквы слова сделать заглавными.

2. В каждом втором предложении поменять местами первое и последнее слово. Задать цвет этих слов в зависимости от количества букв (умножив соответственно на 20 и проверив на допустимость значения).

3. Перетасовать параграфы текста в случайном порядке. Порядок предварительно задать в массиве, вывести нумерацию в сообщении.

4. Удалить слова содержащие букву «а». Слова содержащие «о» покрасить в красный цвет.

5. Слова стоящие в квадратных скобочках заменить на номер слова в скобочках окрашенный в красный цвет. Скобочки оставить.

6. Слова стоящие в квадратных скобочках заменить на длину этого слова, окрасив в зеленый цвет. Скобочки оставить.

7. Слова стоящие в квадратных скобочках заменить на номер параграфа в котором они стоят. Окрасить в зеленый цвет.

8. В словах с двумя буквами «а» и более сделать замену на букву «Л». Окрасить в желтый цвет.

9. В словах с двумя буквами «а» и более сделать все буквы заглавными.

10. В каждом втором слове каждого третьего предложения поменять буквы «а» на символ «@» если буква «а» есть, если ее нет, то все слово заменить на символ «@».

11. В каждом третьем слове поменять местами первую и вторую буквы, а букву в центре слова или две буквы в центре слова в зависимости от четности сделать заглавными и синими.

12. В каждом параграфе сделать по середине вставку количества символов в параграфе, окрасив эту вставку в разные цвета.

13. Удалить в каждом третьем слове букву «а», в каждом втором слове заменить букву «е» на символ «*». Перед символом «*» окрасить символ в любой цвет и увеличить.

14. Пронумеровать все слова в тексте, вставив перед словом его номер. Номер сделать случайным цветом.

15. В тексте все запятые заменить на символы «<>». Внутри <> указать номер запятой. Цвет скобочек сделать случайным.

16. В каждом втором слове в котором встречается символ «а» удалить вторую половину слова. Первую половину слова разделить пополам, вставив пробел.

17. В каждом слове с менее чем пятью буквами, все буквы покрасить в случайные цвета.

18. В каждом втором слове первую половину покрасить в случайные цвета и случайно с вероятностью 0.5 сделать заглавной букву или жирной.

19. В каждой нечетной букве каждого второго слова сделать замену на номер нечетной буквы в слове. Каждый нечетный номер должен быть покрашен сначала в красный, потом в синий цвет.

20. Все слова с более чем пятью согласными необходимо покрасить в красный цвет.

4. Изучение создания макросов OOO Basic Libre Office Calc.

На таблице Calc размещена числовая таблица (матрица). Написать функцию для расчетов по диапазону ячеек в виде XXNN:YYMM. Например = func(A1:B6). Использовать массив. Напоминание: если передать в функцию переменную как диапазон, то это будет двумерный массив границы которого определяются функциями LBound и UBound. Например UBound(Range, 1), где первый параметр указывает на переданный диапазон (массив) и второй параметр на номер измерения, 1 – строки, 2 – столбцы. Таким образом, если 1, то Lbound – верхняя граница, UBound – нижняя, если 2, то левая и правая. По умолчанию, левая и нижняя границы равны 1.

1. Посчитать сумму столбца по середине матрицы и строки по середине. Если матрица с четным числом строк или столбцов брать две строки или два столбца.

2. Посчитать сумму двух диагоналей матрицы.

3. Посчитать сумму верхней и нижней строки и одной диагонали.

4. Посчитать среднее нижней строки и левого столбца.

5. Посчитать сумму верхней строки и главной диагонали.

6. Посчитать произведение суммы главной диагонали на верхний левый и на нижний правый элементы матрицы.

7. Посчитать сумму разниц четных и нечетных элементов диагонали. Матрица с четной размерностью по строкам и столбцам. Например, (2 на 2).

8. Посчитать сумму нижней строки и главной диагонали. И умножить на сумму побочной диагонали.

9. Посчитать сумму всех четных элементов матрицы по столбцам и строкам. (2,2), (2, 4) и т. д.

10. Посчитать сумму верхней и нижней строки. Умножить на сумму диагонали.

11. Посчитать сумму элементов в матрице по номерам чисел Фибоначчи. Нумерация идет от верхней строки последовательно, потом по второй строке и так далее.

Например 1, 2, 3

4, 5, 6 , считать элементы под номером 1, 2, 3, 5, 8 и т. д.

12. Посчитать сумму элементов в матрице по номерам степени двойки. Нумерация идет от верхней строки последовательно, потом по второй строке и так далее.

Например 1, 2, 3

4, 5, 6 , считать элементы под номером 1, 2, 4, 8, 16 и т. д.

13. Посчитать сумму отрицательных элементов матрицы и умножить на сумму положительных элементов.

14. Посчитать среднее значение по матрице и посчитать сумму элементов первой строки больше среднего.

15. Посчитать сумму произведений четных и нечетных строк.

16. Посчитать сумму произведений четных и нечетных столбцов.

17. Посчитать произведение сумм диагональных элементов в матрице. Например:

1 4 5 7

1 2 4 5

6 7 8 1

2 3 4 2

1*(1+4)*(6+2+5)*(2+7+4+7)*(3+8+5)*(4+1)*2

18. Посчитать сумму всех элементов матрицы, умноженную на верхний правый элемент и на среднее значение диагонали.

19. Посчитать сумму среднего по столбцу и по строке элемента матрицы, верхнего правого, левого нижнего.

20. Посчитать сумму всех элементов матрицы не превосходящих правый нижний элемент.