Министерство образования и науки Российской Федерации
Муромский институт (филиал)
федерального государственного бюджетного образовательного учреждения
высшего образования
**«Владимирский государственный университет
имени Александра Григорьевича и Николая Григорьевича Столетовых»
(МИ ВлГУ)**

Вычислительная техника

Методические указания по выполнению лабораторных работ для студентов специальности 11.02.01 «Радиоаппаратостроение»

Текстовое электронное издание

Учебно-методический центр МИ (филиала) ВлГУ
Муром 2017

Составитель:
Смирнов М.С., старший преподаватель кафедры радиотехники МИ (филиала) ВлГУ;

Ответственный за выпуск:
заведующий кафедрой радиотехники,
доктор технических наук, профессор Ромашов Владимир Викторович

Вычислительная техника: Методические указания по выполнению лабораторных работ / сост. Смирнов М.С. [Электронный ресурс]. - Электрон, текстовые дан. (1,4 Мб). - Муром.: МИ (филиал) ВлГУ, 2017. – 1 электрон, опт.диск (CD-ROM). - Систем, требования: процессор x86 с тактовой частотой 500 МГц и выше; 512 Мб ОЗУ; WindowsXP/7/8; видеокарта SVGA 1280x1024 HighColor (32 bit); привод CD-ROM. - Загл. с экрана.

Методические указания содержат сведения, необходимые для выполнения лабораторных работ по дисциплине «Вычислительная техника» для студентов направления 11.02.01 «Радиоаппаратостроение». Тематика работ направлена на приобретение студентами знаний и навыков, необходимых для освоения общепрофессиональных и специальных дисциплин.

Текстовое электронное издание

Минимальные системные требования:
Компьютер: процессор x86 с тактовой частотой 500 МГц и выше; ОЗУ 512 Мб; 5 Мб на жестком диске;
видеокарта SVGA 1280x1024 HighColor (32 bit);
привод CD-ROM
Операционная система: WindowsXP/7/8
Программное обеспечение: AdobeAcrobatReader версии 6 и старше.

Оглавление

Лабораторная работа №1

Расчет количества информации и кодирование данных.

Цель работы:

1. Изучить принципы вероятностного расчет количества информации.
2. Приобретение навыков использования первичного и вторичного алфавита.
3. Приобретение навыков в кодировании и декодировании информации.

Постановка задачи

1. На основе задания построить кодовые последовательности и рассчитать избыточность кода.

Содержание отчета

1. Постановка задачи для конкретного варианта.
2. Подробное описание всех операций.

Теоретические сведения

Единицы измерения информации

Единицы измерения информации служат для измерения объёма информации – величины, исчисляемой логарифмически.

Объём информации можно представлять как логарифм количества возможных состояний.

Наименьшее целое число, логарифм которого положителен – это 2. Соответствующая ему единица – бит – является основой исчисления информации в цифровой технике.

Целые количества бит отвечают количеству состояний, равному степеням двойки.

Особое название имеет 2^2=4 бита – **ниббл** (полубайт, тетрада, четыре двоичных разряда), которые вмещают в себя количество информации, содержащейся в одной шестнадцатеричной цифре.

Байт

Следующей по порядку популярной единицей информации является 2^3=8 бит, или *байт*. Именно к байту (а не к биту) непосредственно приводятся все большие объёмы информации, исчисляемые в компьютерных технологиях.

Таблица 1 Измерения в байтах ГОСТ 8.417-2002

Название	Символ	Степень
байт	Б	2^0
килобайт	КБ	2^{10}
мегабайт	МБ	2^{20}
гигабайт	ГБ	2^{30}
терабайт	ТБ	2^{40}
петабайт	ПБ	2^{50}
эксабайт	ЭБ	2^{60}
зеттабайт	ЗБ	2^{70}
йоттабайт	ЙБ	2^{80}

Трит – логарифмическая единица измерения в теории информации, минимальная целая единица измерения информации источников с тремя равновероятными сообщениями. Энтропию в 1 трит имеет источник информации с тремя равновероятными состояниями. Применяется в теории информации.

1 трит равен $\log_2 3$ бит ≈ 1,585 бит.

По аналогии с понятием Байт существует понятие Трайт. Впервые термин использовался в ЭВМ троичной логики Сетунь-70, где он равнялся 6 тритам.

Дит (Хартли) – логарифмическая единица измерения в теории информации, минимальная целая единица измерения информации источников с десятью равновероятными сообщениями. Энтропию в 1 дит имеет источник информации с десятью равновероятными состояниями. Применяется в теории информации.1 дит равен $\log_2 10$ бит ≈ 3.322 бит.

Нат – одна из единиц измерения информации. Определяется через натуральный логарифм, в отличие от других единиц, где основание логарифма является целым числом. Нат равен $\log_2 e$≈1,443 бит.

Количество информации

Для измерения информации используются различные подходы и методы, например, с использованием меры информации по Р. Хартли и К. Шеннону.

Количество информации – число, адекватно характеризующее разнообразие (структурированность, определенность, выбор состояний и т.д.) в оцениваемой системе. Количество информации часто оценивается в битах, причем такая оценка может выражаться и в долях бит (так как речь идет не об измерении или кодировании сообщений).

Мера информации – критерий оценки количества информации. Обычно она задана некоторой неотрицательной функцией, определенной на множестве событий и являющейся аддитивной, то есть мера конечного объединения событий (множеств) равна сумме мер каждого события.

Формула Хартли

Имеется алфавит А, из букв которого составляется сообщение:

$| A | = m$

Количество возможных вариантов разных сообщений:

$N = m^n$

Где: N – возможное количество различных сообщений, шт; m – количество букв в алфавите, шт; n – количество букв в сообщении, шт.

Пример: Алфавит состоит и 2-х букв В и А, длина сообщения 3 буквы – таким образом, m=2, n=3. При выбранном алфавите и длине сообщения можно составить $N = m^n = 2^3 = 8$ разных сообщений "ВВВ", "ВВА", "ВАВ", "ВАА, "АВВ", "АВА", "ААВ", "ААА" – других вариантов нет.

Задание.

Рассчитать в системе Mathcadколичество возможных сообщений (N).

Таблица 2 Задание для выполнения

№	Количество букв в алфавите	Количество букв в сообщении
1.	5	2,3,4
2.	10	5,6,7
3.	6	6,7,8
4.	11	8,9,10
5.	7	10,11,12
6.	12	13,2,3
7.	8	15,3,8
8.	13	5,19,6
9.	9	12,7,25
10.	14	12,6,11
11.	15	2,9,4
12.	23	10,2,3
13.	16	12,18,16
14.	24	14,5,17
15.	17	18,9,6

Формула Хартли определяется:

$$I = \log_k N = n \cdot \log_k m$$

Где: I-количество информации , k – основание системы счисления.

Уменьшение I говорит об уменьшении разнообразия состояний N системы.

Увеличение I говорит об увеличении разнообразия состояний N системы.

Пример:

Сколько бит, трит, дит и нат занимает слово «информатика»

$$I = 11 \cdot \log_2 33 = 55{,}488 \approx 56 \text{ бит}$$
$$I = 11 \cdot \log_3 33 = 35{,}009 \approx 36 \text{ трит}$$
$$I = 11 \cdot \lg 33 = 16{,}704 \approx 17 \text{ дит}$$
$$I = 11 \cdot \ln 33 = 38{,}642 \approx 39 \text{ нат}$$

Формула Шеннона

Мера Хартли подходит лишь для идеальных, абстрактных систем, так как в реальных системах состояния системы неодинаково осуществимы (неравновероятны).

Существует множество ситуаций, когда возможные события имеют различные вероятности реализации. Например, если монета несимметрична (одна сторона тяжелее другой), то при ее бросании вероятности выпадения "орла" и "решки" будут различаться.

Формулу для вычисления количества информации в случае различных вероятностей событий предложил К. Шеннон в 1948 году. В этом случае количество информации определяется по формуле:

$$I = -\sum_{i=1}^{N} p_i \cdot \log_2 p_i$$

где I – количество информации;
N – количество возможных событий;
p_i – вероятность i-го события.

Увеличение меры Шеннона свидетельствует об уменьшении энтропии (увеличении порядка) системы. Уменьшение меры Шеннона свидетельствует об увеличении энтропии (увеличении беспорядка) системы.

Положительная сторона формулы Шеннона – ее отвлеченность от смысла информации. Кроме того, в отличие от формулы Хартли, она учитывает различность состояний, что делает ее пригодной для практических вычислений. Основная отрицательная сторона формулы Шеннона – она не распознает различные состояния системы с одинаковой вероятностью.

Этот подход к определению количества информации называется *вероятностным*. Подобным образом рассчитывается количество информации в сообщениях. Для этого учитывается частота появления в сообщении тех или иных букв алфавита.

Таблица 3 Частота появления букв кириллического алфавита

Буква	Частота	Буква	Частота	Буква	Частота	Буква	Частота
пробел	0,175	о	0,090	е, ё	0,072	а	0,062
и	0,062	т	0,053	н	0,053	с	0,045
р	0,040	в	0,038	л	0,035	к	0,028
м	0,026	д	0,025	п	0,023	у	0,021
я	0,018	ы	0,016	з	0,016	ъ, ь	0,014
б	0,014	г	0,013	ч	0,012	й	0,010
х	0,009	ж	0,007	ю	0,006	ш	0,006
ц	0,004	щ	0,003	э	0,003	ф	0,002

Таблица 4 Частота появления букв латинского алфавита

Буква	Частота	Буква	Частота	Буква	Частота
пробел	0,175	**e**	0,127	**t**	0,0906
a	0,0817	**o**	0,0751	**i**	0,0697
n	0,0675	**s**	0,0633	**h**	0,0609
r	0,0599	**d**	0,0425	**l**	0,0403
c	0,0278	**u**	0,0276	**m**	0,0241
w	0,0236	**f**	0,0223	**g**	0,0202
y	0,0197	**p**	0,0193	**b**	0,0149
v	0,0098	**k**	0,0077	**x**	0,0015
j	0,0015	**q**	0,001	**z**	0,0007

Пример. Сколько бит, трит, дит и нат занимает слово «информатика»

Расчет количества информации

$p_1 := 0.175 \quad p_2 := 0.09 \quad p_3 := 0.072 \quad p_4 := 0.062 \quad p_5 := 0.062 \quad p_6 := 0.053 \quad p_7 := 0.053$

$p_8 := 0.075 \quad p_9 := 0.04 \quad p_{10} := 0.038 \quad p_{11} := 0.035 \quad p_{12} := 0.028 \quad p_{13} := 0.026 \quad p_{14} := 0.025$

$p_{15} := 0.023 \quad p_{16} := 0.021 \quad p_{17} := 0.018 \quad p_{18} := 0.016 \quad p_{19} := 0.016 \quad p_{20} := 0.014 \quad p_{21} := 0.014$

$p_{22} := 0.013 \quad p_{23} := 0.012 \quad p_{24} := 0.010 \quad p_{25} := 0.009 \quad p_{26} := 0.007 \quad p_{27} := 0.006 \quad p_{28} := 0.006$

$p_{29} := 0.004 \quad p_{30} := 0.003 \quad p_{31} := 0.003 \quad p_{32} := 0.003$

$Ib := -\sum_{i=1}^{32} (p_i \cdot \log(p_i, 2)) \cdot 11 \qquad It := -\sum_{i=1}^{32} (p_i \cdot \log(p_i, 3)) \cdot 11$

$Ib = 48.874$ бит $\qquad It = 30.836$ трит

$Id := -\sum_{i=1}^{32} (p_i \cdot \log(p_i, 10)) \cdot 11 \qquad In := -\sum_{i=1}^{32} (p_i \cdot \ln(p_i)) \cdot 11$

$Id = 14.713$ дит $\qquad In = 33.877$ нат

Задание.

Рассчитать в системе Mathcadкакое количество информации (бит, трит, дит и нат) занимают сообщения с использованием русского и латинского алфавитов.Рассчитать при помощи формул Хартли и Шеннона.

№	Сообщение с использованием русского алфавита	Сообщение с использованием латинского алфавита
1.	автономная система	autonomous systems
2.	проводимость в точке возбуждения	drivingpoint
3.	радионавигационное оборудование	radio as to navigation
4.	содействиенавигации.	aid to navigation
5.	центрированиеповертикали	vertical centering
6.	частотная модуляция	frequencymodulation
7.	подстройка контуров	alignment of tuned circuit
8.	алюминиевый сплав	aluminium allow
9.	радиолокационный высотомер	radar altimeter
10.	усиление низкой частоты	low-frequency amplification
11.	полосовой усилитель	bandpass amplifier
12.	модулированная амплитуда	modulatedamplitude
13.	гетеродинный анализатор гармоник	heterodyne harmonic analyzer
14.	телесный угол прихода радиоволн	acceptance radio angle
15.	ёмкость антенны	antennacapacitance

Алфавит, с помощью которого представляется информация до преобразования является***первичным***; алфавит конечного представления – ***вторичным***.

Код – (1) правило, описывающее соответствие знаков или их сочетаний одного алфавита знакам или их сочетаниям другого алфавита; – (2) знаки вторичного алфавита, используемые для представления знаков или их сочетаний первичного алфавита.

Кодирование – перевод информации, представленной посредством первичного алфавита, в последовательность кодов.

Декодирование– операция, ***обратная*** кодированию, т.е. восстановление информации в первичном алфавите по полученной последовательности кодов.

Операции кодирования и декодирования называются ***обратимыми,*** если их последовательное применение обеспечивает возврат к исходной информации без каких-либо ее потерь.

Если первичный алфавит **A** содержит **N** знаков со средней информацией на знак, определенной с учетом вероятностей их появления, I_1^A. Вторичный алфавит **B** пусть содержит **M** знаков со средней информационной емкостью I_1^B. Пусть также исходное сообщение, представленное в первичном алфавите, содержит **n** знаков, а закодированное сообщение – **m** знаков. Если исходное сообщение содержит $\mathbf{I^{(A)}}$ информации, а закодированное – $\mathbf{I^{(B)}}$, то условие обратимости кодирования, т.е. неисчезновения информации при кодировании, очевидно, может быть записано следующим образом:

$$I^A \le I^B,$$

смысл которого в том, что операция обратимого кодирования может увеличить количество формальной информации в сообщении, но не может его уменьшить. Однако каждая из величин в данном неравенстве может быть заменена произведением числа знаков на среднюю информационную емкость знака, т.е.:

$$n \cdot I_1^A \le m \cdot I_1^B$$

или

$$I_1^A \le \frac{m}{n} \cdot I_1^B$$

Пример. Рассчитать минимальное количество символов латинского алфавита необходимое для кодирования слова «информатика» транслитом. Таблицы вероятности появления букв русского и латинского алфавита приведены в таблицах 1-2 .

$$IB := -\sum_{i=1}^{27} \left(t_i \cdot \log\left(t_i, 2\right)\right) = 4.072 \qquad IA := -\sum_{i=1}^{33} \left(p_i \cdot \log\left(p_i, 2\right)\right) = 4.408$$

информатика

$$n := 11$$

$$m0 := \frac{n \cdot IA}{IB} = 11.907$$

$$m := 12$$

$$K := \frac{m}{n} = 1.091$$

$$Q := 1 - \frac{IA}{K \cdot IB} = 0.008$$

Рисунок 1.1 Расчет m, K и Q

Отношение **m/n**, характеризует среднее число знаков вторичного алфавита, которое приходится использовать для кодирования одного знака первичного алфавита и называется ***длиной кодовой цепочки***– $\mathbf{K^{(B)}}$ (верхний индекс указывает алфавит кодов).

В частном случае, когда появление любых знаков вторичного алфавита равновероятно, согласно формуле Хартли$I_1^B = \log_2 M$, и тогда

$$\frac{I_1^A}{\log_2 M} \le K^B (1)$$

Избыточность кода **(Q)**показывает, насколько операция кодирования увеличила длину исходного сообщения:

$$Q = 1 - \frac{I^A}{I^B} = 1 - \frac{I_1^A}{K^B \cdot I_1^B} (2)$$

Чем меньше **Q** (т.е. чем ближе она к 0 или, что то же, $\mathbf{I^{(B)}}$ ближе к $\mathbf{I^{(A)}}$), тем более выгодным оказывается код и более эффективной операция кодирования.

Далее в основном ограничимся ситуацией, когда **M = 2**, т.е. для представления кодов в линии связи используется лишь два типа сигналов – "0" и "1". Удобство двоичных кодов и в том, что при равных длительностях и вероятностях каждый элементарный сигнал (0 или 1) несет в себе 1 бит информации **($\log_2$M = 1)**; тогда из выражения (1):

$$I_1^A \le K^{(2)}$$

Применение формулы (2) для двоичного кодирования дает:

$$Q = 1 - \frac{I_1^A}{K^{(2)}}$$

Определение количества переданной информации при двоичном кодировании сводится к простому подсчету числа единиц и нулей. При этом возникает проблема выделения из потока сигналов отдельных кодов. Приемное устройство фиксирует интенсивность и длительность сигналов. Элементарные сигналы (0 и 1) могут иметь одинаковые или разные длительности. Их количество в коде (длина кодовой цепочки), который ставится в соответствие знаку первичного алфавита, также может быть одинаковым (в этом случае код называется равномерным) или разным (неравномерный код). Наконец, коды могут строиться для каждого знака исходного алфавита (алфавитное кодирование) или для их комбинаций (кодирование блоков, слов). В результате при кодировании (алфавитном и словесном) возможны следующие варианты сочетаний:

Таблица 5Вариантысочетаний

Длительности элементарных сигналов	Кодировка первичных символов (слов)	Ситуация
одинаковые	равномерная	(1)
одинаковые	неравномерная	(2)
разные	равномерная	(3)
разные	неравномерная	(4)

В случае использования неравномерного кодирования или сигналов разной длительности (ситуации (2), (3) и (4)) для отделения кода одного знака от другого между ними необходимо передавать специальный сигнал – временной разделитель (признак конца знака) или применять такие коды, которые оказываются уникальными, т.е. несовпадающими с частями других кодов. При равномерном кодировании одинаковыми по длительности сигналами (ситуация (1)) передачи специального разделителя не требуется, поскольку отделение одного кода от другого производится по общей длительности, которая для всех кодов оказывается одинаковой (или одинаковому числу бит при хранении).

Длительность двоичного элементарного импульса (τ) показывает, сколько времени требуется для передачи 1 бит информации. Очевидно, для передачи информации, в среднем приходящейся на знак первичного алфавита, необходимо время $K^{(2)} \cdot \tau$. Таким образом, задачу оптимизации кодирования можно сформулировать в иных терминах: построить такую систему кодирования, чтобы суммарная длительность кодов при передаче (или суммарное число кодов при хранении) данного сообщения была бы наименьшей.

Алфавитное неравномерное двоичное кодирование

Данный случай относится к варианту (2). При этом как следует из названия, символы некоторого первичного алфавита (например, русского) кодируются комбинациями символов двоичного алфавита (т.е. 0 и 1), причем, длина кодов и, соответственно, длительность передачи отдельного кода, могут различаться. Длительности элементарных сигналов при этом одинаковы ($\tau_0 = \tau_1 = \tau$). За счет чего можно оптимизировать кодирование в этом случае? Очевидно, суммарная длительность сообщения будет меньше, если применить следующий подход: **тем буквам первичного алфавита, которые встречаются чаще, присвоить более короткие по длительности коды, а тем, относительная частота которых меньше – коды более длинные**.

Но длительность кода – величина дискретная, она кратна длительности сигнала τпередающего один символ двоичного алфавита. Следовательно, коды букв, вероятность появления которых в сообщении выше, следует строить из возможно меньшего числа элементарных сигналов. Построим кодовую таблицу для букв русского алфавита, основываясь на приведенных ранее вероятностях появления отдельных букв.

Очевидно, возможны различные варианты двоичного кодирования, однако, не все они будут пригодны для практического использования – важно, чтобы закодированное сообщение могло быть однозначно декодировано, т.е. чтобы в последовательности 0 и 1, которая представляет собой многобуквенное кодированное сообщение, всегда можно было бы различить обозначения отдельных букв. Проще всего этого достичь, если коды будут разграничены разделителем – некоторой постоянной комбинацией двоичных знаков. Условимся, что разделителем отдельных кодов букв будет последовательность 00 (признак конца знака), а разделителем слов – 000 (признак конца слова – пробел). Довольно очевидными оказываются следующие правила построения кодов:

1. код признака конца знака может быть включен в код буквы, поскольку не существует отдельно (т.е. кода всех букв будут заканчиваться 00);
2. коды букв не должны содержать двух и более нулей подряд в середине (иначе они будут восприниматься как конец знака);
3. код буквы (кроме пробела) всегда должен начинаться с 1;
4. разделителю слов (000) всегда предшествует признак конца знака; при этом реализуется последовательность 00000 (т.е. если в конце кода встречается комбинация …000 или …0000, они не воспринимаются как разделитель слов); следовательно, коды букв могут оканчиваться на 0 или 00 (до признака конца знака).

Длительность передачи каждого отдельного кода t_i, очевидно, может быть найдена следующим образом: $t_i = k_i \cdot \tau$, где k_i – количество элементарных сигналов (бит) в коде символа i. В соответствии с приведенными выше правилами формируются следующие таблицы кодов:

Таблица 6Таблицакодоврусского алфавита

Буква	Код	$p_i \cdot 10^3$	k_i	Буква	Код	$p_i \cdot 10^3$	k_i
пробел	000	174	3	**я**	1011000	18	7
о	100	90	3	**ы**	1011100	16	7
е	1000	72	4	**з**	1101000	16	7
а	1100	62	4	**ь,ъ**	1101100	14	7
и	10000	62	5	**б**	1110000	14	7
т	10100	53	5	**г**	1110100	13	7
н	11000	53	5	**ч**	1111000	12	7
с	11100	45	5	**й**	1111100	10	7
р	101000	40	6	**х**	10101000	9	8
в	101100	38	6	**ж**	10101100	7	8
л	110000	35	6	**ю**	10110000	6	8
к	110100	28	6	**ш**	10110100	6	8
м	111000	26	6	**ц**	10111000	4	8
д	111100	25	6	**щ**	10111100	3	8
п	1010000	23	7	**э**	11010000	3	8
у	1010100	21	7	**ф**	11010100	2	8

Теперь по формуле можно найти среднюю длину кода $\mathbf{K^{(2)}}$ для данного способа кодирования:

$$K^{(2)} = \sum_{i=1}^{32} p_i \cdot k_i = 4.964$$

Поскольку для русского языка, $\mathbf{I_1^{(r)}}$**=4,356** бит, избыточность данного кода, согласно (2), составляет:

$$Q^R = 1 - \frac{4.356}{4.964} \approx 0.122$$

это означает, что при данном способе кодирования будет передаваться приблизительно на 12% больше информации, чем содержит исходное сообщение.

Таблица 7 Таблица латинского русского алфавита

Буква	Код	$p_i \cdot 10^3$	k_i	Буква	Код	$p_i \cdot 10^3$	k_i
пробел	000	191	3	**m**	1010000	20	7
e	100	104	3	**f**	1010100	20	7
t	1000	73	4	**w**	1011000	17	7
a	1100	65	4	**g**	1011100	16	7
o	10000	60	5	**y**	1101000	15	7
n	10100	56	5	**p**	1101100	14	7
i	11000	56	5	**b**	1110000	12	7
s	11100	52	5	**v**	1110100	8	7
r	101000	50	6	**k**	1111000	5	7
h	101100	59	6	**x**	1111100	1	7
d	110000	35	6	**j**	10101000	1	8
l	110100	33	6	**q**	10101100	1	8
u	111000	23	6	**z**	10110000	1	8
c	111100	22	6				

Теперь по формуле можно найти среднюю длину кода $\mathbf{K^{(2)}}$ для данного способа кодирования:

$$K^{(2)} = \sum_{i=1}^{27} p_i \cdot k_i = 4.716$$

Поскольку для английского языка, $\mathbf{I_1^{(e)}}$**=4,072** бит, избыточность данного кода, согласно (2), составляет:

$$Q^E = 1 - \frac{4.072}{4.716} \approx 0.137$$

Пример. Закодировать при помощи неравновесного алфавитного кодирования слово «информатика» и определить количество байт необходимое для хранения данного кода.

При помощи таблицы 2, составляем код:

и	н	ф	о	р	м	а	т	и	к	а
10000	11000	11010100	100	101000	111000	1100	10100	10000	110100	1100

Количество бит, после кодирования – 57.
Минимальное количество байт для хранения – 8.

Префиксные коды.

При рассмотрении вариантов двоичного неравномерного кодирования возникает проблема наиболее оптимальный способ неравномерного двоичного кодирования без использования разделительных знаков.

Суть проблемы состоит в нахождении такого варианта кодирования сообщения, при котором последующее выделение из него каждого отдельного знака (т.е. декодирование) оказывается однозначным без специальных указателей разделения знаков. Наиболее простыми и употребимыми кодами такого типа являются так называемые префиксные коды, которые удовлетворяют следующему условию (условию Фано):

Неравномерный код может быть однозначно декодирован, если никакой из кодов не совпадает с началом (префиксом) какого-либо иного более длинного кода.

Например, если имеется код 110, то уже не могут использоваться коды 1, 11, 1101, 110101 и пр. Если условие Фано выполняется, то при прочтении (расшифровке) закодированного сообщения путем сопоставления со списком кодов всегда можно точно указать, где заканчивается один код и начинается другой.

Использование префиксного кодирования позволяет делать сообщение более коротким, поскольку нет необходимости передавать разделители знаков. Однако условие Фано не устанавливает способа формирования префиксного кода и, в частности, наилучшего из возможных.

Код Хаффмана.

Способ оптимального префиксного двоичного кодирования был предложен ***Д.* Хаффманом**.

Пример.Имеется первичный алфавит **A**, состоящий из шести знаков с вероятностями появления в сообщении, a_1=0,3; a_2=0,2; a_3=0,2; a_4=0,15; a_5=0,1; a_6=0,05. Первоначально создается новый вспомогательный алфавит $\mathbf{A^{(1)}}$, при помощи объединения двух знаков с наименьшими вероятностями (a_5 и a_6), и заменой их одним знаком (например, $\mathbf{a^{(1)}}$); вероятность нового знака будет равна сумме вероятностей тех, что в него вошли, т.е. 0,15; остальные знаки исходного алфавита включаются в новый без изменений; общее число знаков в новом алфавите, очевидно, будет на 1 меньше, чем в исходном.

Аналогичным образом создаются новые алфавиты, пока в последнем не останется два знака; ясно, что число таких шагов будет равно $N-2$, где ***N*** – число знаков исходного алфавита (в нашем случае **N=6**, следовательно, необходимо построить 4 вспомогательных алфавита). В промежуточных алфавитах каждый раз будем переупорядочивать знаки по убыванию вероятностей.

Всю процедуру построения можно представить в виде таблицы:

№ знака	Вероятности				
	Исходный алфавит	Промежуточные алфавиты			
		$A^{(1)}$	$A^{(2)}$	$A^{(3)}$	$A^{(4)}$
1	0,3	0,3	0,3	0,4	0,6
2	0,2	0,2	0,3	0,3	0,4
3	0,2	0,2	0,2	0,3	
4	0,15	0,15	0,2		
5	0,1	0,15			
6	0,05				

Теперь в обратном направлении проводится процедура кодирования. Двум знакам последнего алфавита присваиваются коды 0 и 1 (которому какой – роли не играет; например, что верхний знак будет имеет код 0, а нижний – 1). То есть, знак $\mathbf{a_1^{(4)}}$ алфавита $\mathbf{A^{(4)}}$, имеющий вероятность 0,6 , получит код 0, а $\mathbf{a_2^{(4)}}$ с вероятностью 0,4 – код 1. В алфавите $\mathbf{A^{(3)}}$ знак $\mathbf{a_1^{(3)}}$ с вероятностью 0,4 сохранит свой код (1); коды знаков $\mathbf{a_2^{(3)}}$ и $\mathbf{a_3^{(3)}}$, объединенных знаком $\mathbf{a_1^{(4)}}$ с вероятностью 0,6 , будут уже двузначным: их первой цифрой станет код связанного с ними знака (т.е. 0), а вторая цифра – как условились – у верхнего 0, у нижнего – 1; таким образом, $\mathbf{a_2^{(3)}}$ будет иметь код 00, а $a_3^{(3)}$ – код 01. Полностью процедура кодирования представлена в следующей таблице:

№ знака	Вероятности									
	Исходный алфавит		Промежуточные алфавиты							
			$A^{(1)}$		$A^{(2)}$		$A^{(3)}$		$A^{(4)}$	
1	0,3	00	0,3	00	0,3	00	0,4	1	0,6	**0**
2	0,2	10	0,2	10	0,3	01	0,3	00	0,4	**1**
3	0,2	11	0,2	11	0,2	10	0,3	01		
4	0,15	010	0,15	010	0,2	11				
5	0,1	0110	0,15	011						
6	0,05	0111								

Из самой процедуры построения кодов легко видеть, что они удовлетворяют условию Фано и, следовательно, не требуют разделителя. Средняя длина кода при этом оказывается:

$$K^{(2)} = 0{,}3 \cdot 2 + 0{,}2 \cdot 2 + 0{,}2 \cdot 2 + 0{,}15 \cdot 3 + 0{,}1 \cdot 1 + 0{,}05 \cdot 5 = 2{,}45$$

Для сравнения можно найти $\mathbf{I_1^{(A)}}$.

$$I_1^A = 0{,}3 \cdot \log_2 0{,}3 + 0{,}2 \cdot \log_2 0{,}2 + 0{,}2 \cdot \log_2 0{,}2 + 0{,}15 \cdot \log_2 0{,}15 + 0{,}1 \cdot \log_2 0{,}1 + 0{,}05 \cdot \log_2 0{,}05 = 2{,}409$$

Таким образом, избыточность кода равна:

$$Q = 1 - \frac{2.409}{2.45} = 0.0169 \approx 1.7\%$$

Код Хаффмана важен в теоретическом отношении, поскольку можно доказать, что он является самым экономичным из всех возможных, т.е. ни для какого метода алфавитного кодирования длина кода не может оказаться меньше, чем код Хаффмана.

Применение описанного метода для букв русского алфавита дает следующие коды:

Таблица 8 Таблица кодов русских букв

Буква	Код	$p_i \cdot 10^3$	k_i	Буква	Код	$p_i \cdot 10^3$	k_i
пробел	000	174	3	**я**	0011001	18	6
о	111	90	3	**ы**	0101100	16	6
е	0100	72	4	**з**	010111	16	6
а	0110	62	4	**ь,ъ**	100001	14	6
и	0111	62	4	**б**	101100	14	6
т	1001	53	4	**г**	101101	13	6
н	1010	53	5	**ч**	110011	12	6
с	1101	45	4	**й**	0011001	10	7
р	00101	40	5	**х**	1000000	9	7
в	00111	38	5	**ж**	1000001	7	7
л	01010	35	5	**ю**	1100101	6	7
к	10001	28	5	**ш**	00110000	6	8
м	10111	26	5	**ц**	11001000	4	8
д	11000	25	5	**щ**	11001001	3	8
п	001000	23	6	**э**	001100010	3	9
у	001001	21	6	**ф**	001100011	2	9

Средняя длина кода оказывается равной $\mathbf{K^{(2)} = 4{,}395}$; избыточность кода $\mathbf{Q^{(r)} = 0{,}00887}$, т.е. менее 1%.

Задания на работу

Задание №1.1 Согласно варианту в таблице 7, рассчитать минимальное количество символов латинского алфавита необходимое для кодирования слов русского алфавита.

Задание №1.2 Согласно варианту в таблице 7, рассчитать минимальное количество символов латинского алфавита необходимое для кодирования слов русского алфавита.

Задание 2. Закодировать при помощи неравновесного алфавитного кодирования словосочетание согласно заданию в таблице 7 и определить количество байт необходимое для хранения данного кода.

Задание 3. Составить префиксный код Хаффмана для алфавита, приведенного в таблице 8. Для алфавита вычислить среднюю длину кодаK, среднееколичество информации$I_1^{(A)}$, и избыточность кода Q.

Таблица 9 Таблица заданий

№	Сообщение с использованием русского алфавита	Сообщение с использованием латинского алфавита
16.	автономная система	autonomous systems
17.	проводимость в точке возбуждения	drivingpoint
18.	радионавигационное оборудование	radio as to navigation
19.	содействиенавигации.	aid to navigation
20.	центрированиеповертикали	vertical centering
21.	частотная модуляция	frequencymodulation
22.	подстройка контуров	alignment of tuned circuit
23.	алюминиевый сплав	aluminium allow
24.	радиолокационный высотомер	radar altimeter
25.	усиление низкой частоты	low-frequency amplification
26.	полосовой усилитель	bandpass amplifier
27.	модулированная амплитуда	modulatedamplitude
28.	гетеродинный анализатор гармоник	heterodyne harmonic analyzer
29.	телесный угол прихода радиоволн	acceptance radio angle
30.	ёмкость антенны	antennacapacitance

Таблица 10 Алфавиты для получения кода Хаффмана

№	Вероятности символов алфавита A								
	a_1	a_2	a_3	a_4	a_5	a_6	a_7	a_8	a_9
1.	0,25	0,2	0,2	0,15	0,1	0,04	0,03	0,02	0,01
2.	0,01	0,2	0,06	0,15	0,13	0,18	0,1	0,05	0,12
3.	0,3	0,1	0,1	0,01	0,01	0,07	0,12	0,2	0,09
4.	0,02	0,4	0,01	0,02	0,1	0,14	0,13	0,15	0,03
5.	0,5	0,05	0,15	0,15	0,01	0,03	0,07	0,02	0,02
6.	0,16	0,14	0,05	0,17	0,21	0,09	0,05	0,1	0,03
7.	0,02	0,16	0,3	0,2	0,01	0,14	0,15	0,01	0,01
8.	0,01	0,01	0,05	0,08	0,1	0,15	0,2	0,2	0,2
9.	0,01	0,01	0,01	0,01	0,47	0,46	0,01	0,01	0,01
10.	0,3	0,4	0,02	0,04	0,12	0,03	0,06	0,01	0,02
11.	0,4	0,03	0,08	0,16	0,09	0,1	0,01	0,1	0,03
12.	0,05	0,02	0,09	0,2	0,12	0,13	0,21	0,08	0,1
13.	0,24	0,21	0,06	0,12	0,15	0,04	0,06	0,07	0,05
14.	0,12	0,31	0,09	0,11	0,03	0,02	0,2	0,09	0,03
15.	0,25	0,11	0,09	0,01	0,03	0,15	0,2	0,14	0,02

Лабораторная работа№2
Моделирование логических блоков ЭВМ

Цель работы:

1. Научиться моделировать и использовать комбинаторные логические элементы.
2. Научиться моделировать и использовать элементы запоминающих устройств.

Постановка задачи

1. Используя программный пакет MicroCap создать модели комбинаторных устройств и провести временной анализ данных моделей.
2. Используя программный пакет MicroCap создать модели синхронных запоминающих устройств и провести временной анализ данных моделей.

Содержание отчета

1. Постановка задачи для конкретного варианта.
2. Скриншот модели.
3. Осциллограммы реализации временного анализа.

Теоретические сведения

1. Моделирование мультиплексора.

Мультиплексор представляет собой схемный элемент, который подключает один из цифровых входов к одному выходу. Вход выбирается из группы посредством селективных (адресных) входов.

В качестве примера представлен 4-разрядный мультиплексор (рис. 1).

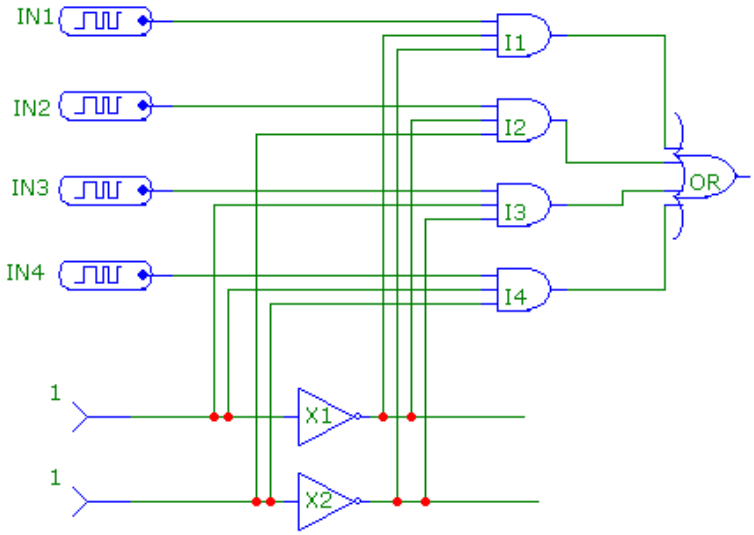


Рисунок 2.1 4-входовый мультиплексор

Мультиплексор имеет в своем составе:

- ✓ 4 информационных входа (IN1…IN4);
- ✓ 2 адресных входа (X1, X2);

Каждый информационный вход мультиплексора соединен через элемент AND(I1…I4) с адресными входами X1 и X2. Сигнал с адресных входов снимается до и после инвертора. Работа мультиплексора описывается следующим образом:

1. Чтобы выбрать вход IN1,на адресные входы X1 и X2 необходимо подать комбинацию 00. Вход IN1 соединен с сигналами ⌐X1, ⌐X2(X1 и X2 после инвертора). Таким образом, на вход I1 придет сигнал IN1 и две логические единицы с X1 и X2. На вход элемента ORс I1 придет сигнал IN1, а с I2…I4 придут логические нули. На выходе элемента ORбудет только сигнал IN1.
2. Чтобы выбрать вход IN2,на адресные входы X1 и X2 необходимо подать комбинацию 01. Вход IN2соединен с сигналами ⌐X1, X2. Таким образом, на вход I2придет сигнал IN2и две логические единицы с X1 и X2. На вход элемента ORс I2придет сигнал IN2, а с I1, I3, I4 придут логические нули. На выходе элемента ORбудет только сигнал IN2.

3. Чтобы выбрать вход IN3,на адресные входы X1 и X2 необходимо подать комбинацию 10. Вход IN3соединен с сигналами X1,¬X2. Таким образом, на вход I3придет сигнал IN3и две логические единицы с X1 и X2. На вход элемента ORс I3 придет сигнал IN3, а с I1, I2, I4 придут логические нули. На выходе элемента ORбудет только сигнал IN3.
4. Чтобы выбрать вход IN4,на адресные входы X1 и X2 необходимо подать комбинацию 11. Вход IN4соединен с сигналами X1,X2. Таким образом, на вход I4придет сигнал IN4и две логические единицы с X1 и X2. На вход элемента ORс I4 придет сигнал IN4, а с I1, I2, I3 придут логические нули. На выходе элемента ORбудет только сигнал IN4.

Задание.

Реализовать 8-входовый мультиплексор с тремя адресными входамии входом CS.Если CS=1,то мультиплексор работает в штатном режиме. Если CS=0,то на выходе мультиплексора всегда 0.

2. Моделирование счетчика.

Счетчиками в цифровой технике называются специальные элементы, позволяющие подсчитывать число поступивших на вход импульсов. Понятие «счетчик импульсов» тесно связано с понятием «делитель частоты». По сути дела это одно и то же устройство.

В качестве простейшего делителя частоты может выступать D-триггер (рис. 2). Для того, что бы перевести D-триггер в счетный режим, нужно соединить инверсный выход триггера Q с его D входом. Теперь, если на вход CLK подать импульсный сигнал некоторой постоянной частоты, то по спаду каждого входного импульса триггер будет переключаться в противоположное состояние. В результате на выходе D-триггера мы получим другой сигнал с частотой следования импульсов в два раза меньшей, чем частота импульсов на его входе.

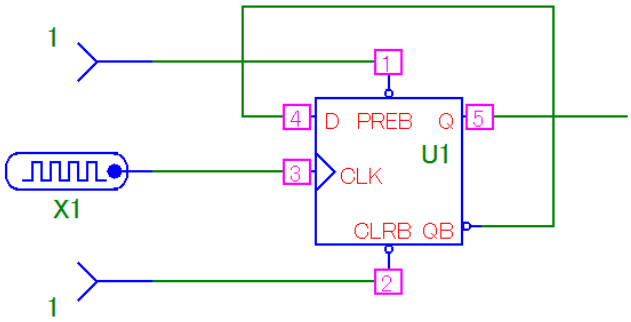


Рисунок 2.2 Двоичный счетчик

Задание.

Создать делитель частоты с выхода которого снимаются сигналы частотой 10МГц.

Делители широко используются в цифровой технике. Цепочка последовательно соединенных D-триггеров, позволяет получить сигналы, требуемой частоты, путем деления импульсов задающего генератора. Например, соединенные последовательно два делителя, позволят получить сигнал с частотой в четыре раза меньшей, чем входная. Три каскада делителя дадут деление на восемь. Четыре – на шестнадцать. И так далее. На рис. 2.3 изображен четырехкаскадный делитель частоты на D-триггерах.

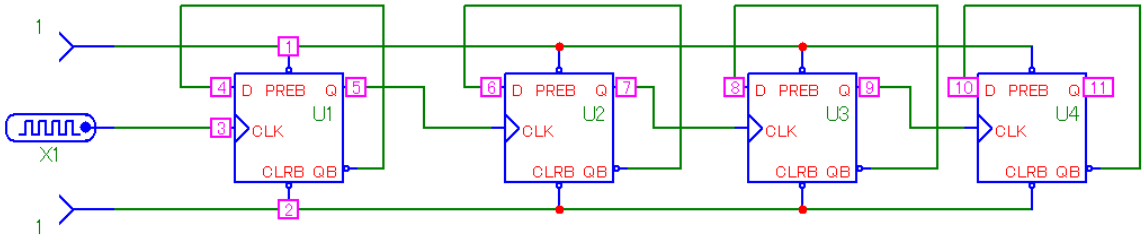


Рисунок 2.3 Четырехкаскадный делитель частоты на D-триггерах

Импульсы тактового генератора поступают на вход первого каскада деления. Выход первого каскада подключен к входу второго и т.д. Если частота сигнала на входе равна f, то на выходах делителя мы получим сигналы со следующими частотами:

5 узел =*f*/2;
7 узел =*f*/4;
9 узел =*f*/8
11 узел =*f*/16

Приведенную на рис. 2.3 схему можно использовать не только в качестве делителя частоты, но и в качестве счетчика входных импульсов. Представьте, что выходы 5,7,9,11 – это разряды некоторого двоичного числа. 5 – младший разряд, а 11 – самый старший. При поступлении на вход схемы счетных импульсов, D-триггера, входящие в схему, будут переключаться так, как это было показано ранее. Двоичное число, образованное выходными разрядами делителя, также будет меняться. Предположим, что перед приходом самого первого импульса на всех четырех выходах схемы были логические нули. Тогда, при поступлении входных импульсов, двоичное число будет принимать следующие значения (см. табл. 1).

11	9	7	5	Десятичный эквивалент
0	0	0	0	0
0	0	0	1	1
0	0	1	0	2
0	0	1	1	3
0	1	0	0	4
0	1	0	1	5
0	1	1	0	6
0	1	1	1	7
1	0	0	0	8
1	0	0	1	9
1	0	1	0	10
1	0	1	1	11
1	1	0	0	12
1	1	0	1	13
1	1	1	0	14
1	1	1	1	15

Моделирование элементов памяти.

1. Моделирование RS-триггера.

Схему асинхронного триггера можно представить как логическую схему, у которой, по крайней мере, один из выходов соединен с входом. Триггеры называют также последовательностными схемами или конечными автоматами. Поведение триггера зависит как от значений входных переменных в данный момент времени, так и от входных переменных, в предыдущие моменты времени. Поэтому он может хранить информацию.

В качестве примера проведена модель RS-триггера собранная на элементах ИЛИ-НЕ (NOR). RS-триггер представляет собой идеальную логическую схему с обратной связью (рис. 2.4). Сокращения S и R, которыми обозначаются входные сигналы,означают «установка» (set) и «возврат» (reset). Здесь два выхода– прямой (Q2) и инвертирующий (Q1).

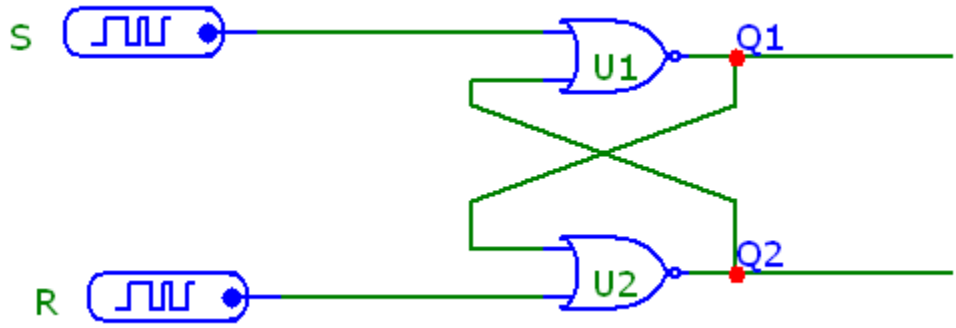


Рисунок 2.4 Модель RS-триггера на элементах NOR.

Выходные значения будут зависеть как от входных сигналов, так и от предыдущего состояния триггера:

1. **S=1, R=0**. В этом случае сигнал на выходеверхнего NOR-вентиля имеет значение **Q1=0**. Сигналына обоих входах нижнего NOR-вентиля имеют значения 0, такчто **Q2=1**.
2. В противоположном случае, когда **S=0, R=1**, вследствиесимметрии устанавливаются значения на выходах **Q2=0** и **Q1=1**.
3. **S=0, R=0**.В этом случаеповедение триггера будет определяться предыдущим состоянием.Если выходной сигнал **Q2=1**, то входной сигналверхнего вентиля равен 1 и сохраняется **Q1=0**. Сохраняетсятакже **Q2=1**, так как сигналы на обоих входах этого вентиляимеют значение 0. Данное состояние стабильно и поэтомуудерживается.Если на выходе **Q1=1**изсоображений симметрии удерживаются **Q2=0** и **Q1=1**.
4. **S=1, R=1**. Вэтом случае оба выхода устанавливаются на 0. Этот вариант исключается, так как выходы не будут взаимно инверсными.

Таким образом, можно построить таблицу истинности для RSтриггера на элементах NOR.

Таблица 11Таблица истинности RS-триггера

S	R	Q1	Q2
0	0	¬Q	Q
1	0	1	0
0	1	0	1
1	1	–	–

Задание. Построить модель RS-триггера на элементах И-НЕ(NAND). Построить таблицу истинности для этого триггера.

2. Моделирование RS-триггера с тактовым входом.

Главной проблемой для RS-триггеров является то, что появлениедаже коротких импульсов помехи на входах R и S ведет к ошибочнойустановке и возврату в исходное состояние триггера. Поэтомупри работе cRS-триггерами, применяют тактовый сигнал, определяющий момент времени, в течение которого выходы триггера могут быть установлены в новое состояние.

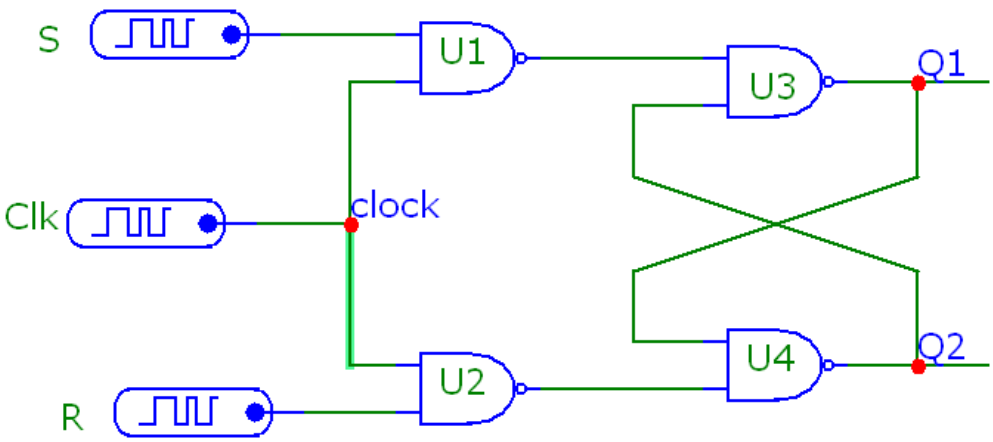


Рисунок 2.5 RS-триггер с тактовым входом

Триггер на рисунке 2.5 может быть запущен только при положительном тактовом импульсе (Clk=1). Этот вид управления называют управлениемуровнем тактового импульса или управлением состоянием.

Система моделирования Microcap имеет в своем составе модель RS-триггера с управлением тактовым импульсом (рис.2.6).

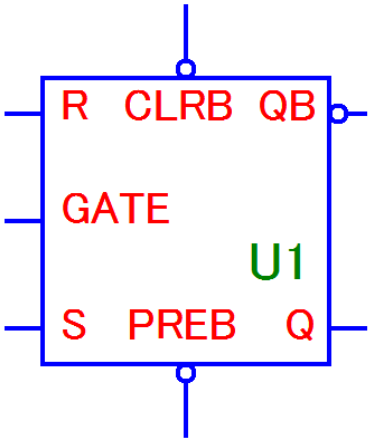


Рисунок 2.6 Модель RS-триггера.

Для подключения модели в схему необходимо использовать компонент DigitalPrimitives→GatesFlip-Flops→SRFF:

Sи **R** – входы Setи Reset;

Qи **QB** – прямой и инвертирующий выходы триггера;

GATE – вход тактового сигнала. При поступлении логической «1» на данный вход триггер переключается в новое состояние.

PREB – вход начальной установки триггера. На данный вход подавать логическую единицу для начальной установки;

CLRB –вход асинхронного сброса триггера. Если на это вход приходит логическая «1» триггер сбрасывается в начальное состояние.

Задание. Смоделировать схему, представленную на рисунке 2.5 и построить ее таблицу истинности. Построить таблицу истинности для готовой модели и сравнить ее с таблицей синтезированногоRS-триггера.

3. Моделирование D-триггера.

Основной недостаток RS-триггера, заключается в наличии запрещенного входного состояния. Для устранения этого недостатка применяют другие триггеры. Важнейшим является D-триггер (рис. 2.7).

D-триггер образован RS-триггером с управлением тактовым уровнем, в котором для нового входа D установлено, что D=S=¬R.

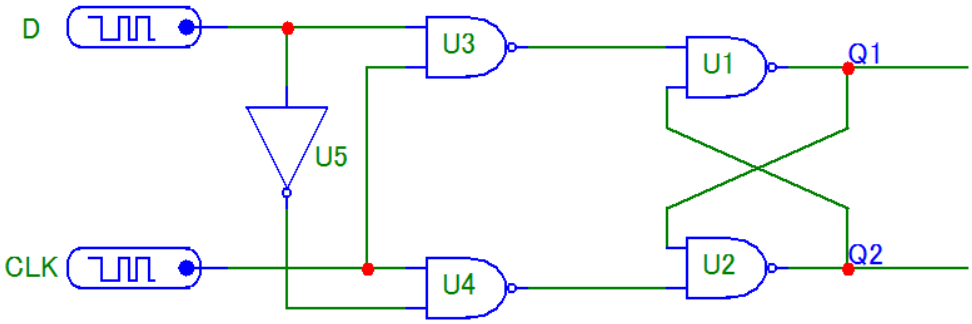


Рисунок 2.7 Логическая схема D-триггера

Передаточная функция D-триггера говорит о том, что при **C=1** вход для ввода данных переключается, а при **C=0** запоминается старое состояние.

Система моделирования Microcaримеет в своем составе модель D-триггера с управлением тактовым импульсом (рис.2.8).

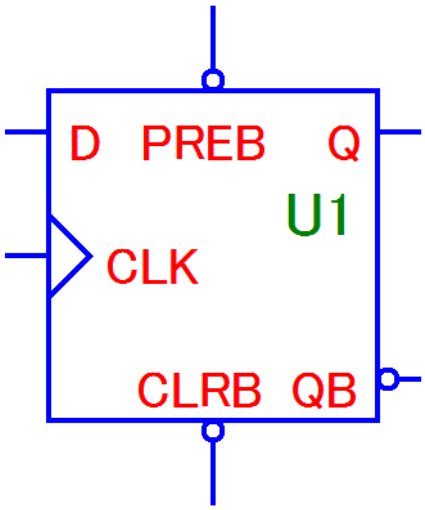


Рисунок 2.8 Модель D-триггера.

Для подключения модели в схему необходимо использовать компонент DigitalPrimitives→Edge-TriggeredFlip-Flop→DFF:

D – входтриггера;

Qи **QB** – прямой и инвертирующий выходы триггера;

CLK– вход тактового сигнала. При поступлении логической «1» на данный вход триггер переключается в новое состояние.

PREB – вход начальной установки триггера. На данный вход подавать логическую единицу для начальной установки;

CLRB –вход асинхронного сброса триггера. Если на это вход приходит логическая «1» триггер сбрасывается в начальное состояние.

Задание. Смоделировать схему, представленную на рисунке 2.8 и построить ее таблицу истинности. Построить таблицу истинности для готовой модели D-триггера и сравнить ее с таблицей синтезированного D-триггера.

4. **Моделирование JK-триггера.**

JK-триггер может быть образован на основе RS-триггера типа master-slave путем подачи обратной связи с выходов Q1 и Q2 на входы R и S при высоком уровне тактового сигнала (рисунок 2.9).

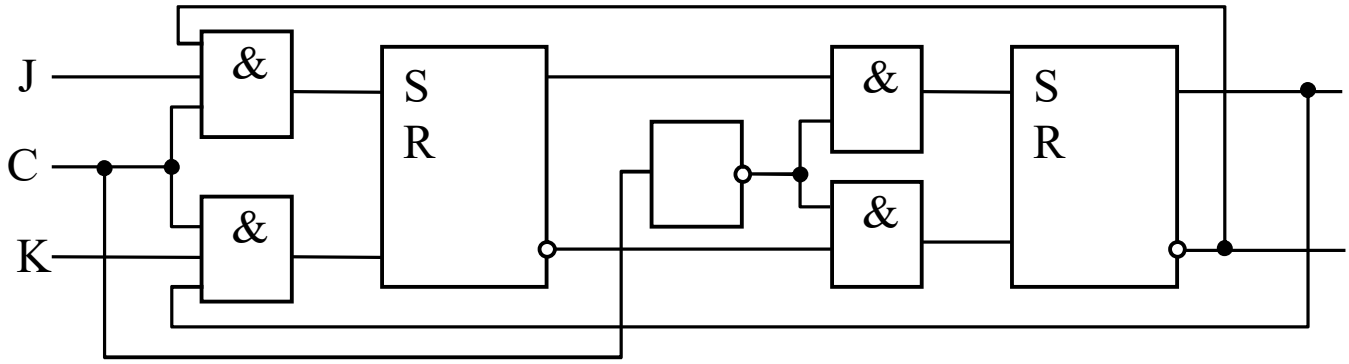


Рисунок 2.9 Логическая схема JK-триггера

Установку триггера можно производитьтолько после его возвращения в исходное состояние. Поскольку R и S не могут быть одновременно равны 1, так как справедливо либо Q1=1 либо Q2 = 1.

JK-триггер ведет себя как RS-master-slave-триггер, пока сигналы J и K не будут одновременно равны 1. Но если**J=K=1** выходной сигнал изменяется при каждом тактовомимпульсе. Это упрощает конструирование делителей частоты и цифровыхсчетчиков на основе JK-триггеров.

Система моделирования Microcap имеет в своем составе модель JK-триггера с управлением тактовым импульсом (рис.2.10).

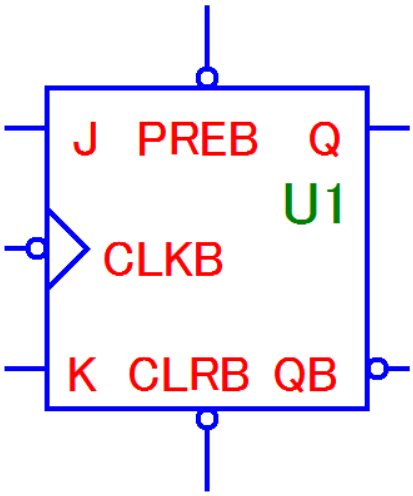


Рисунок 2.10 Модель JK-триггера.

Для подключения модели в схему необходимо использовать компонент Digital Primitives→Edge Triggered Flip-Flop →JKFF:

Ju K – входитриггера;

Qи **QB** – прямой и инвертирующий выходы триггера;

CLK– вход тактового сигнала. При поступлении логической «1» на данный вход триггер переключается в новое состояние.

PREB – вход начальной установки триггера. На данный вход подавать логическую единицу для начальной установки;

CLRB –вход асинхронного сброса триггера. Если на это вход приходит логическая «1» триггер сбрасывается в начальное состояние.

Задание. Смоделировать схему, представленную на рисунке 2.9 и построить ее таблицу истинности. Построить таблицу истинности для готовой модели JK-триггера, и сравнить ее с таблицей синтезированного JK-триггера.

Регистры

Регистр представляет собой упорядоченную последовательность триггеров, число которых соответствует числу разрядов в слове. С каждым регистром обычно связано комбинационное цифровое устройство, с помощью которого обеспечивается выполнение некоторых операций над словами.

Фактически любое цифровое устройство можно представить в виде совокупности регистров, соединённых друг с другом при помощи комбинационных цифровых устройств.

Основой построения регистров являются D-триггеры, RS-триггеры.

Операции в регистрах

Типичными являются следующие операции:

- ✓ приём слова в регистр;
- ✓ передача слова из регистра;
- ✓ поразрядные логические операции;
- ✓ сдвиг слова влево или вправо на заданное число разрядов;
- ✓ преобразование последовательного кода слова в параллельный и обратно;
- ✓ установка регистра в начальное состояние (сброс).

Классификация регистров

Регистры классифицируются по следующим видам:

- ✓ параллельные – запись и считывание информации происходит одновременно на все входы и со всех выходов;
- ✓ последовательные – запись и считывание информации происходит в первый триггер, а та информация, которая была в этом триггере, перезаписывается в следующий – то же самое происходит и с остальными триггерами;
- ✓ комбинированные.

5. Моделирование параллельного регистра

В **параллельном регистре** (рис 2.11) на тактируемых D-триггерах код запоминаемого числа подается на информационные входы всех триггеров и записывается в регистр с приходом тактового импульса. Выходная информация изменяется с подачей нового входного слова и приходом следующего синхроимпульса. Такие регистры используют в системах оперативной памяти. Число триггеров в них равно максимальной разрядности хранимых слов.

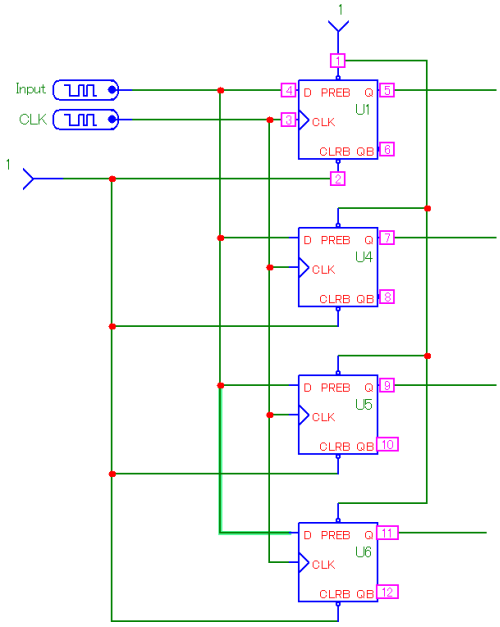


Рисунок 2.11 Параллельный 4-разрядный регистр

В приведенном параллельном 4-разрядном регистре на все информационные входы подается сигнал с одного генератора. Изменения в триггерах происходят одновременно с подачей на вход CLKтактового сигнала.

Задание.

Реализовать 8-разрядный параллельный регистр.

Входной сигнал на регистр должен подаваться с 8-разрядного генератора.

В регистре предусмотреть сигналы:

- ✓ CLR (для сброса содержимого регистра).
- ✓ CS – сигнал для выбора микросхемы. Новая информация в регистр может заноситься только в том случае, если CS=1.

Сдвигающие (последовательные) регистры

Последовательные (сдвигающие) регистры представляют собою цепочку разрядных схем, связанных цепями переноса (рис. 2.12). В однотактных регистрах со сдвигом на один разряд вправо слово сдвигается при поступлении синхросигнала. Вход и выход последовательные (DSR – DataSerialRight).

По приходу тактового импульса первый триггер записывает код (0 или 1), находящийся в этот момент на его входе D, а каждый следующий триггер переключается в состояние, в котором до этого находился предыдущий. Так происходит потому, что записываемый сигнал проходит с входа D триггера к выходу Q с задержкой, большей длительности фронта тактового импульса (в течение которого происходит запись). Каждый тактовый импульс последовательно сдвигает код числа в регистре на один разряд. Поэтому для записи N-разрядного кода требуется N тактов.

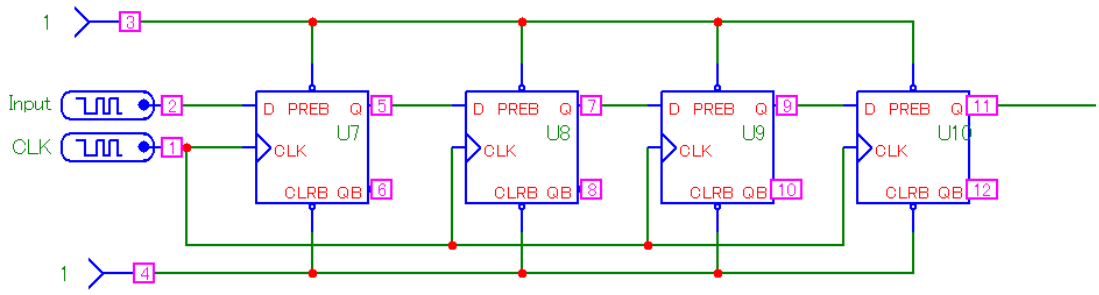


Рисунок 2.12 Последовательный 4-разрядный регистр

Задание.

1. Реализовать 4-разрядный сдвиговый регистр (рис 2.13).

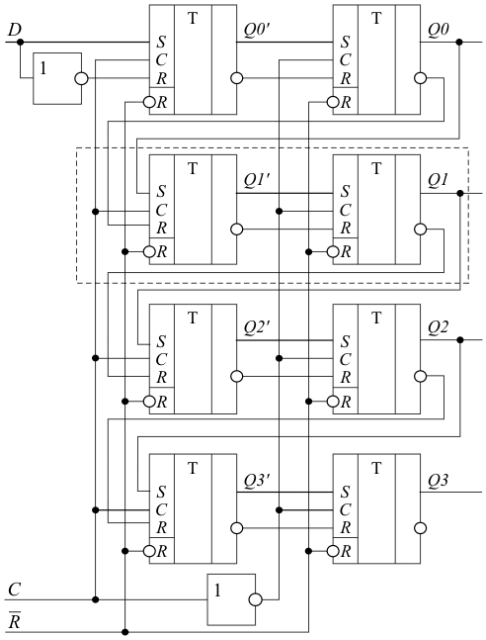


Рисунок 2.13 4-разрядный сдвиговый регистр

2. На основе 8-разрядного последовательного регистра реализовать преобразователь последовательного кода в параллельный.

В регистре предусмотреть сигналы:

- ✓ CLR (для сброса содержимого регистра).
- ✓ CS – сигнал для выбора микросхемы. Новая информация в регистр может заноситься только в том случае, если CS=1.

Лабораторная работа №3
Реализация программ с использованием стандартных исполнителей.

Цели работы.

1. Получение знаний по алгоритмической машине Поста.
2. Приобретение навыков создания программ для алгоритмической машины Поста.

Постановка задачи

1. Создать программы для алгоритмической машины Поста.

Содержание отчета

1. Постановка задачи для конкретного варианта.
2. Подробное описание всех операций.

Теоретические сведения

Машина Поста, как и ее близкий родственник – машина Тьюринга, – представляет собой мысленную конструкцию, существующую лишь в воображении.Таким образом, машины Поста и Тьюринга являются «абстрактными» вычислительными машинами.

Машина Поста состоит из ленты и каретки (называемой также считывающей и записывающей головкой). Лента бесконечна и разделена на секции одинакового размера (рис. 3.1).

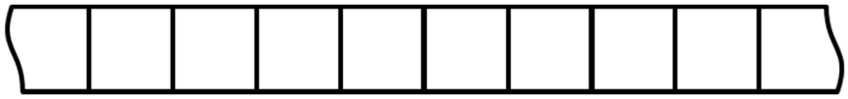


Рисунок 3.1 Лента машины Поста

Порядок, в котором расположены секции ленты, подобен порядку, в котором расположены все целые числа. Поэтому естественно ввести на ленте «целочисленную систему координат», занумеровав секции целыми числами: …-3, -2, -1, 0, 1, 2, 3,... (рис. 3.2).

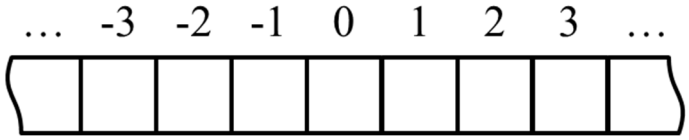


Рисунок 3.2 Нумерованная лента Машины Поста.

Система координат жестко сопоставлена с лентой, что дает возможность указывать какую-либо секцию ленты, называя ее порядковый номер, или координату.

В каждой секции ленты может быть либо ничего не записано (такая секция называется пустой), либо записана метка V (тогда секция называется отмеченной) (рис. 3.3)

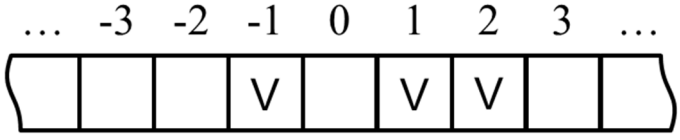


Рисунок 3.3 Лента машины Поста с заполненными метками

Информация о том, какие секции пусты, а какие отмечены, образует состояние ленты. Иными словами, состояние ленты – это распределение меток по ее секциям. Каретка может передвигаться вдоль ленты влево и вправо. Когда ока неподвижна, она стоит против ровно одной секции ленты (рис. 3.4).

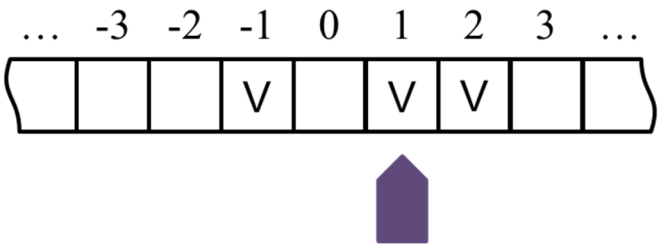


Рисунок 3.4 Машина Поста с кареткой

Информация о том, какие секции пусты, а какие отмечены и где стоит каретка, образует состояние машины Поста.

Таким образом, состояние машины слагается из состояния ленты и указания номера той секции, которую обозревает каретка. За единицу времени (которую мы будем называть шагом) каретка может сдвинуться на одну секцию влево или вправо. Кроме того, каретка может поставить (напечатать) или уничтожить (стереть) метку в той секции, против которой она стоит, а также распознать, стоит или нет метка в обозреваемой ею секции.

Работа машины Поста состоит в том, что каретка передвигается вдоль ленты и печатает или стирает метки, согласно заложенной программе.

Каждая программа машины Поста состоит из команд. Командой машины Поста называется выражение, имеющее один из следующих шести видов (буквы i, j, j_1, j_2 означают натуральные числа 1, 2, 3, 4, 5, …).

Первый вид. Команды движения вправо: $i. \Rightarrow j$. Выполнение команды движения вправо состоит в том, что каретка сдвигается на одну секцию вправо.

Второй вид. Команды движения влево. $i. \Leftarrow j$. Выполнение команды движения влево состоит в том, что каретка сдвигается на одну секцию влево.

Третий вид. Команды печатания метки. $i. \vee j$. Выполнение команды печатания метки состоит в том, что каретка ставит метку на обозреваемой секции; выполнение этой команды возможно лишь в том случае, если обозреваемая перед началом выполнения команды секция пуста; если же на обозреваемой секции уже стоит метка, команда считается невыполнимой.

Четвертый вид. Команды стирания метки. $i. \xi j$. Выполнение команды стирания метки состоит в том, что каретка уничтожает метку в обозреваемой секции; выполнение этой команды возможно лишь в том случае, если обозреваемая секция отмечена; если же на обозреваемой секции метка отсутствует, то команда считается невыполнимой.

Пятый вид. Команды передачи управления.

$$i. \ ? \begin{array}{l} \nearrow j_1 \\ \searrow j_2 \end{array}$$

Выполнение команды передачи управления с верхней отсылкой $\boldsymbol{j_1}$ и нижней отсылкой $\boldsymbol{j_2}$ никак не изменяет состояния машины: ни одна из меток не уничтожается и не ставится, и каретка также остается неподвижной (машина делает, так сказать, «шаг на месте»). Если секция, обозреваемая перед началом выполнения этой команды, была пуста, то следующей должна выполняться команда с номером $\boldsymbol{j_1}$, если же эта секция была отмечена, следующей должна выполняться команда с номером $\boldsymbol{j_2}$.

Шестой вид. Команды остановки. $i.$ стоп. Выполнение команды остановки тоже никак не меняет состояния машины и состоит в том, что машина останавливается.

Например, 137. => 1 является командой движения вправо,

$$25. \ ? \begin{array}{l} \nearrow 32 \\ \searrow 25 \end{array}$$

командой передачи управления, а 6386. стоп – командой остановки.

Число $\boldsymbol{i}$, стоящее в начале команды, называется номером команды. Так, у приведенных только что команд номера соответственно 137, 25 и 6386.

Число $\boldsymbol{j}$, стоящее в конце команды (а у команд передачи управления – каждое из чисел $\boldsymbol{j_1}$ и $\boldsymbol{j_2}$), называется отсылкой (при этом в команде передачи управления $\boldsymbol{j_1}$ – верхней, а $\boldsymbol{j_2}$ – нижней отсылкой). У команд остановки нет отсылки. Так, у приведенных только что команд отсылками служат числа 1, 32, 25, причем 32 – верхняя отсылка, а 25 – нижняя отсылка.

Программа машины Поста называется конечный и непустой (т.е. содержащий хотя бы одну команду) список команд машины Поста, обладающий следующими двумя свойствами:

1. На k-м месте стоит команда с номером k, т.е. на первом месте в этом списке стоит команда с номером 1, на втором месте (если оно есть) – команда с номером 2 и т.д.
2. Отсылка любой команд из списка совпадает с номером некоторой (другой или той же самой) команды списка (более точно: для каждой отсылки каждой команды списка найдется в списке такая команда, номер которой равен рассматриваемой отсылке).

Например, следующий список будет программой машины Поста:

1. стоп
2. ? $\begin{matrix}\nearrow 4\\ \searrow 1\end{matrix}$
3. ξ 3
4. стоп

А эти два списка не будут программами машины Поста, хотя и составлены из команд машины Поста

2. ? $\begin{matrix}\nearrow 4\\ \searrow 1\end{matrix}$
3. ξ 3
1. стоп
4. стоп

(не выполнено первое условие);

1. стоп
2. ? $\begin{matrix}\nearrow 4\\ \searrow 5\end{matrix}$
3. ξ 3
4. стоп

(не выполнено второе условие).

Чтобы машина Поста начала работать, надо задать, во-первых, некоторую программу, а во-вторых, некоторое ее (машины) состояние, т. е. как-то расставить метки по секциям ленты (в частности, можно все секции оставить пустыми) и поставить каретку против одной из секций. Как правило, предполагается, что в начальном состоянии машины каретка ставится всегда против секции с номером (координатой) нуль. При таком соглашении начальное состояние машины полностью определено состоянием ленты.

Работа машины на основании заданной программы (и при заданном начальном состоянии) происходит следующим образом. Машина приводится в начальное состояние и приступает к выполнению первой команды программы. Эта команда выполняется за один шаг, после чего машина приступает к выполнению той команды, номер которой (назовем его $\boldsymbol{\alpha}$) равен отсылке (одной из отсылок, если их две) первой команды. Эта команда также выполняется за один шаг, после чего начинается выполнение команды, номер которой равен отсылке команды с номером $\boldsymbol{\alpha}$.

Вообще каждая команда выполняется за один шаг, а переход от выполнения одной команды к выполнению другой происходит по следующему правилу: пусть на k-м шаге выполнялась команда с номером $\boldsymbol{i}$, тогда, если эта команда имеет единственную отсылку $\boldsymbol{j}$, то на k+1-м шаге выполняется команда с номером $\boldsymbol{j}$; если эта команда имеет две отсылки $\boldsymbol{j_1}$ и, то на k+1-м шаге выполняется одна из двух команд – с номером $\boldsymbol{j_1}$ или с номером $\boldsymbol{j_2}$; если, наконец, выполняющаяся на k-м шаге команда вовсе не имеет отсылки, то на k+1-м шаге и на всех последующих шагах не выполняется никакая команда: машина останавливается.

Если теперь, задавшись программой и каким-либо начальным состоянием, пустить машину в ход, то осуществится один из следующих трех вариантов:

1) В ходе выполнения программы машина дойдет до выполнения невыполнимой команды (печатания метки в непустой секции или стирания метки в пустой секции); выполнение программы тогда прекращается, машина останавливается; происходит так называемая безрезультатная остановка.

2) В ходе выполнения программы машина дойдет до выполнения команды остановки; программа в этом случае считается выполненной, машина останавливается; происходит так называемая результативная остановка.
3) В ходе выполнения программы машина не дойдет до выполнения ни одной из команд, указанных в первых двух вариантах; выполнение программы при этом никогда не прекращается, машина никогда не останавливается; процесс работы машины происходит бесконечно.

На машине Поста можно совершать разнообразные вычисления. Самое простейшее из таких вычислений – прибавление единицы к произвольному числу. Это вычисление осуществляется в нескольких вариантах, зависящих от той или иной формулировки начальных условий.

Рассмотрение того, как на машине Поста производится прибавление единицы – позволяет познакомить понять проблематику, возникающую и при оперировании с реальными быстродействующими вычислительными машинами. Дело в том, что основная математическая задача, встающая перед человеком при работе его на вычислительных машинах, остается той же самой и для реальных и для «абстрактных» машин. Эта задача состоит в составлении для машины программы, приводящей к заданной цели: составление таких программ называется программированием.

Выполнение заданий № 1-3.

Запись чисел в машине Поста и постановка задачи о прибавлении единицы

На машине Поста можно производить различные действия над числами. Речь будет всегда идти о ряде натуральных чисел 0, 1, 2, 3, 4,…

Рассмотрим конечную последовательность идущих подряд друг за другом отмеченных секций ленты, заключенную между двумя пустыми секциями. Такую последовательность отмеченных секций будем называть массивом, а число секций в ней – длиной массива. Так, на рис. 3.5 показаны массивы длиной 3, 5, 1 и 2.

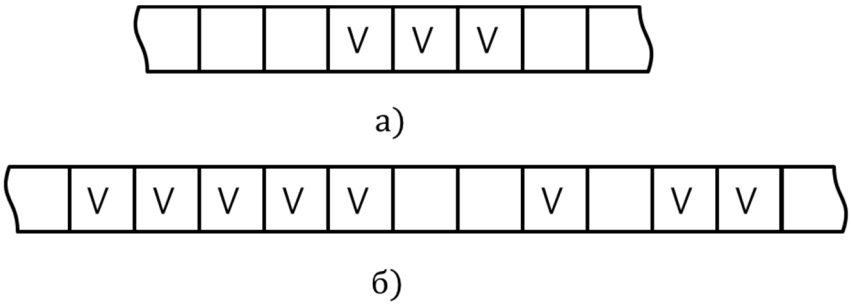


Рисунок 3.5 Массивы, представленные в машине Поста

Нумерация элементов любого массива всегда начинается с 0. Таким образом, число ***n*** записывается на ленте посредством массива длины ***n***+1, а сам этот массив называется машинной записью числа ***n***. На рис. 5 показаны, следовательно, машинные записи чисел 2 (рис. 3.5а), 4, 0 и 1 (рис. 3.5б).

Зададимся целью осуществить на машине Поста прибавление единицы. Надо составить программу, которая, будучи применена к ленте, на которой записано произвольное число ***n*** приводила бы к результативной остановке, причем после остановки на ленте должно быть записано число ***n***+1 (имеется в виду составление одной программы, годящейся для любого ***n***).

В предыдущей фразе задача еще поставлена не вполне точно, так как ни о начальном (т. е. задаваемом в начале), ни о заключительном (т.е. возникающем после результативной остановки) состоянии машины не сказано, нигде именно записано число, ни что еще записано на ленте; не сказано также, где должна стоять каретка. Будем предполагать, что в начале и в конце работы программы на ленте имеются только записи соответствующих чисел (***n*** в начале и ***n***+1 в конце), расположенные в произвольном месте ленты, а в остальном лента пуста. В целях облегчения нашей задачи не будем налагать никаких дополнительных ограничений на заключительное состояние: нас, таким образом, устроит любая программа, приводящая к ленте с записью числа ***n***+1, где бы эта запись ни находилась и где бы ни стояла при этом каретка. В то же время мы будем делать все более и более широкие

предположения относительно взаимного расположения каретки и машинной записи числа в начальном состоянии машины.

Прибавление единицы в простейшем случае

Мы начнем с самого сильного ограничения, налагаемого на взаимное расположение машинной записи числаи каретки в начале, и, тем самым, с наиболее простой задачи. Назовем ее задачей 1.

Задача 1 (длинная формулировка). Требуется написать программу машины Поста, обладающую следующим свойством. Каково бы ни было число ***n***, если начальное состояние машины Поста таково, что на ленте имеется машинная запись числа ***n*** (а в остальном лента пуста) и каретка стоит против самой левой секции записи, то выполнение программы должно привести к результативной остановке, после чего на ленте (в произвольном ее месте) должно быть записано число ***n***+1 (а в остальном лента должна быть пуста), причем каретка может стоять где угодно.

Обозначим через A_n совокупность всех таких состояний машины Поста, в каждом из которых отмеченными на ленте являются равно ***n***+1 секций, а каретка стоит против самой левой из отмеченных секций. На рис. 3.6 показано несколько состояний из класса A_2; они отличаются друг от друга лишь различным положением массива и «жестко связанной с ним» каретки относительно начала координат.

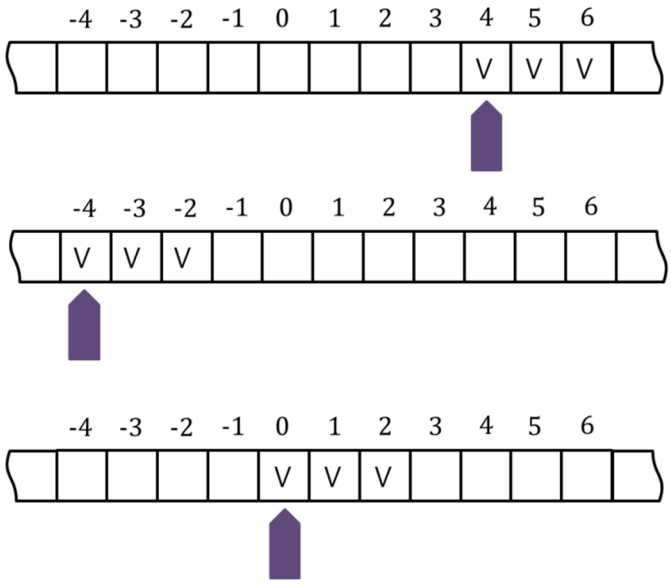


Рисунок 3.6 Состояния машины Поста класса A_2

Обозначим через E_n совокупность всех таких состояний машины Поста, в каждом из которых отмеченными на ленте являются ровно ***n***+1 секций, а каретка может стоять где угодно. К E_n относятся все состояния из A_nи еще много других. На рис. 16 показано несколько состояний из класса E_2.

Теперь задачу 1 можно сформулировать короче.

Задача 1 (короткая формулировка). Написать такую программу машины Поста, которая для любого ***n*** будучи применена к произвольному состоянию из класса A_n, дает результативную остановку в каком-то состоянии из класса E_{n+1}.

Прежде чем приступить к решению задачи 1, заметим, что в ее условии ничего не говорится о том, что должно получаться при применении искомой программы к состояниям, не принадлежащим ни к одному из классов A_n (***n***=0, 1,2, ...); это значит, что нам безразлично, что будет происходить при таких состояниях,выбираемых в качестве начальных: нас устроит любая программа, переводящая состояния из A_n в состояния из E_{n+1}, что бы она ни делала с остальными состояниями.

Решением задачи 1 будет, например, такая программа:

Программа I_1

1. ⇐ 2
2. V 3
3. стоп

Мы написали «например», потому что решение задачи 1 не единственно: возможны и другие программы, удовлетворяющие условиям задачи. Например, такая программа также будет решением задачи 1:

1. ⇒2

2. ? → 3 / 3

3. ⇐ 3
4. ⇐ 3
5. ∨6
6. стоп

Однако написанная выше программа I_1 будет самой короткой из программ, удовлетворяющих условиям задачи 1.

Совершенно аналогично задаче 1 может быть решена задача 1', отличающаяся от задачи 1 лишь тем, что вначале каретка обозревает самую правую из секций массива. Обозначим через A'_n класс всех таких состояний из $\boldsymbol{E_n}$, в которых каретка обозревает самую правую из отмеченных секций. Тогда короткая формулировка задачи 1' будет такова:

Задача 1' (короткая формулировка). Написать такую программу машины Поста, которая для любого $\boldsymbol{n}$, будучи применена к произвольному состоянию из класса A'_n, дает результативную остановку в каком-то состоянии из класса $\boldsymbol{E_{n+1}}$.

ПрограммаI'$_1$

1. ⇒ 2
2. ∨ 3
3. стоп

Прибавление единицы в более сложных случаях

Не будем теперь требовать, чтобы каретка в начале стояла непременно против одной из крайних секций массива, как это было в задачах 1 и 1'. Ограничимся требованием, чтобы в начальном состоянии каретка обозревала одну из секций массива.

Задача 2 (длинная формулировка). Требуется написать программу машины Поста, обладающую следующим свойством. Каково бы ни было число $\boldsymbol{n}$, если начальное состояние машины Поста таково, что на ленте имеется машинная запись числа $\boldsymbol{n}$ (а в остальном лента пуста) и каретка стоит против одной из секций записи, то выполнение программы должно привести к результативной остановке, после чего на ленте (в произвольном ее месте) должно быть записано число $\boldsymbol{n}$+1 (а в остальном лента должна быть пуста), причем каретка может стоять где угодно.

Обозначим через $\boldsymbol{B_n}$ совокупность всех таких состояний машины Поста, в каждом из которых отмеченными на ленте являются ровно $\boldsymbol{n}$+1 секций, а каретка стоит против одной из отмеченных секций. Очевидно, класс $\boldsymbol{B_n}$ является частью класса $\boldsymbol{E_n}$ и сам содержит в качестве части класс $\boldsymbol{A_n}$. На рис. 3.7 изображено несколько состояний из класса $\boldsymbol{B_4}$. Тогда получаем следующую короткую формулировку задачи 2:

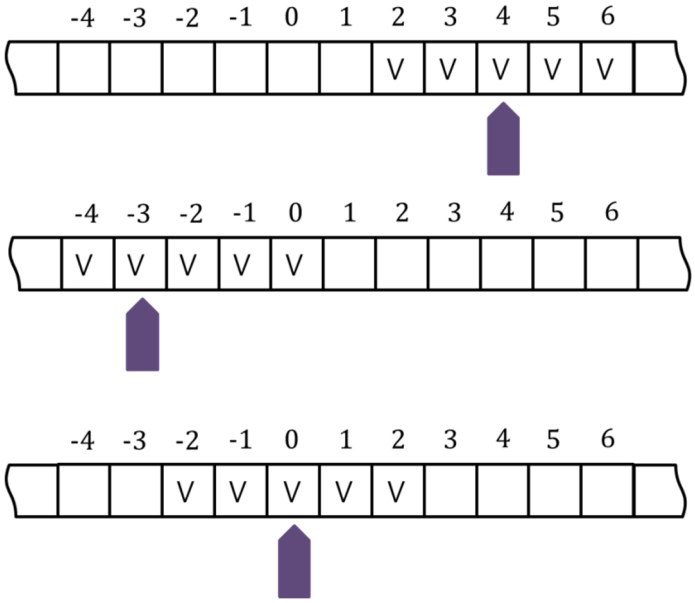


Рисунок 3.7 Состояния машины Поста класса B4

Задача 2 (короткая формулировка). Написать такую программу машины Поста, которая для любого n, будучи применена к произвольному состоянию из класса B_n, дает результативную остановку в каком-то состоянии из класса E_{n+1}.

Очевидно, что каждое решение задачи 2 будет одновременно и решением задачи 1, а также задачи 1'. В то же время существуют решения задачи 1, не являющиеся решениями задачи 2. Такова, например, программа I_1. Так же, как и для задачи 1, для задачи 2 существует бесконечное множество программ, являющихся ее решением. Мы по-прежнему будем интересоваться наиболее короткими программами. Можно доказать, что никакая программа длины 1, 2 или 3 не может быть решением задачи 2. В то же время возможны решения задачи 2 длины 4, Вот одно из таких решений:

Программа II_1

1. ⇐ 2
2. ? → 3, 1
3. ∨4
4. стоп

Упражнение 1. Найдите еще два решения задачи 2, имеющих длину 4 и содержащих команду движения влево. Проверьте, что каждое из найденных вами решений содержит команду печатания метки, команду передачи управления и команду остановки.

Упражнение 2. Докажите, что, помимо программы II_1, существует еще ровно одиннадцать (II_2, II_3,…,II_{12}) решений задачи 2, имеющих длину 4 и содержащих команду движения влево. Выпишите эти решения.

Упражнение 3. Проверьте, что программы II'_1,II'_2,…, II'_{12}, получающиеся из программ II_1, II_2,…,II_{12} путем замены знака ⇐ на знак ⇒, также служат решениями задачи 2. Докажите, что программами II_1, II_2,…,II_{12}и II'_1,II'_2,…, II'_{12} исчерпываются все решениязадачи 2 длины 4.

Рассмотрим теперь такие начальные состояния, в которых каретка держит в поле зрения не какую-то из секций исходного массива, а какую-то пустую секцию. Будем предполагать при этом, что является заранее известным, стоит ли каретка слева или справа от исходного массива. Слова «является заранее известным» означают, что требуется найти не единую программу, работающую во всех случаях, а две программы, одна из которых работает в случае, если каретка вначале стоит слева от массива, а другая работает в случае, когда каретка вначале стоит справа от массива. Таким образом, мы имеем здесь не одну, а две задачи. Чтобы получить сразу короткие формулировки этих задач, введем следующие обозначения. Обозначим через C_n совокупность всех таких состояний из класса E_n, в которых каретка стоит слева от массива, а через C'_n– совокупность всех таких состояний из класса E_n в которых каретка стоит справа от массива. На рис. 3.8 показан общий вид состояния из класса C_n, а на рис. 3.9- общий вид состояния из класса C'_n.

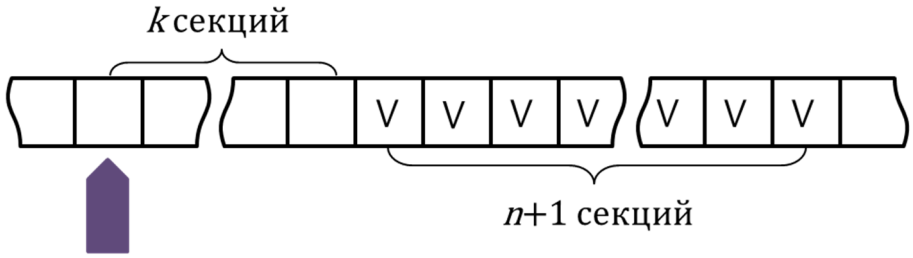


Рисунок 3.8 Общий вид состояния из класса C_n. Здесь $k \geq 0$

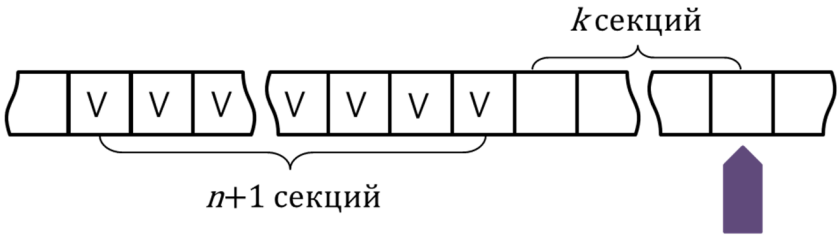


Рисунок 3.9 Общий вид состояния из класса C'_n. Здесь k≥0

Задача 3 (короткая формулировка). Написать такую программу машины Поста, которая для любого n, будучи применена к произвольному состоянию из класса C_n, дает результативную остановку в каком-то состоянии из класса E_{n+1}.

Задача 3' (короткая формулировка). Написать такую программу машины Поста, которая для любого n, будучи применена к произвольному состоянию из класса C'_n, дает результативную остановку в каком-то состоянии из класса E_{n+1}.

Очевидно, что каждую программу, являющуюся решением задачи 3, можно превратить в решение задачи 3', если всюду знак ⇒заменить на знак⇐, а знак ⇐ заменить на знак ⇒. Такой же операцией любое решение задачи 3' превращается в решение задачи 3. Поэтому достаточно решить только одну из этих задач. Вот одно из решений задачи 3:

Программа III

1. ⇒ 2
2. ? → 1, 3
3. ⇐ 4
4. ∨ 5
5. стоп

Прибавление единицы в еще более сложном случае

В этом параграфе мы рассмотрим в качестве класса исходных (начальных) состояний объединение всех классов B_0, C_0, B_1, C_1, B_2, C_2,… из предыдущего параграфа. Этот класс будет, следовательно, состоять из всех таких – и только таких – состояний машины, в каждом из которых каретка стоит либо против какой-то из секций исходного массива, либо же левее массива.

Обозначим через D_n совокупность всех таких состояний из класса E_n, в которых обозреваемая кареткой секция либо отмечена, либо расположена левее всех отмеченных секций. Очевидно, что при каждом n класс D_n есть объединение классов B_n и C_n.

Задача 4. Написать такую программу машины Поста, которая для любого n, будучи применена к произвольному состоянию из класса D_n, дает результативную остановку в каком-то состоянии из класса E_{n+1}.

Задачу 4 можно свести к задачам 2 и 3, т. е. можно указать способ, как из произвольных решений задач 2 и 3 получить некоторое решение задачи 4. Укажем такой способ. Пусть **Ξ**– произвольный список команд машины Поста, аk — произвольное целое число. Через **Ξ[+k]** будем обозначать список, полученный из **Ξ** путем прибавления числа k ко всем номерам и всем отсылкам команд из**Ξ**. Например, I_1[+7] – это такой список:

8. ⇐ 9
9. ∨ 10
10. стоп

Пусть теперь II — произвольная программа, являющаяся решением задачи 2, а III — произвольная программа, являющаяся решением задачи 3. Пусть l есть длина программы II. Составим теперь такой список команд:

1. ? → l+2, 2

II [+**1**]
III [+l+1]

Легко проверить, что этот список команд является программой машины Поста, причем такой программой, которая удовлетворяет условиям задачи 4. В самом деле, всякое состояние из класса D_n принадлежит либо B_n, либо C_n. В первом случае «срабатывает» список II[+1], во втором – список III[+l+1].

Однако изложенный способ не обязан давать (и, как мы сейчас увидим, действительно не дает) кратчайшее решение задачи 4 – даже если исходить из кратчайших решений задач 2 и 3. Посмотрим, что получится, если применить этот способ к программам II_1 и III. Мы получим такое решение задачи 4:

Программа IV[10]

1. ? → 6, → 2
2. ⇐ 3
3. ? → 4, → 2
4. V5
5. стоп
6. ⇒7
7. ? → 6, → 8
8. ⇐ 9
9. V 10
10. стоп

Ясно, что эту программу можно сократить, не меняя при этом совершаемой ею работы, путем объединения в одну двух команд остановки, или, как мы будем говорить, путем поглощения одной из этих команд другой. Проще поглотить команду № 10 командой № 5. Для этого достаточно заменить отсылку 10 во всех командах, где она встречается, – в нашем случае только в команде № 9 — на отсылку 5 и затем вычеркнуть команду № 10[1].

Мы получим программу IV[9], в которой можно теперь поглотить команду № 9 командой № 4. Для этого достаточно заменить в команде № 8 отсылку 9 на отсылку 4 и затем вычеркнуть команду № 9. После произведенных двух поглощений получим новое решение задачи 4 в виде программы IV[8]

1. ? → 6, → 2
2. ⇐ 3
3. ? → 4, → 2
4. V5
5. стоп
6. ⇒7
7. ? → 6, → 8
8. ⇐4

Замечание. В общем случае поглощение (в данной программе) команды №**α** командой №**β** состоит в последовательном выполнении трех операций. Во-первых, всюду отсылка **α** заменяется на отсылку **β**; во-вторых, команда №**α** вычеркивается; в-третьих, во всех командах образовавшегося списка числа **α+1**, **α+2**, **α+3**, ... (которые могут быть как номерами, так и отсылками) заменяются соответственно на **α**, **α+1**,**α+2**,… Если команды №**α** и №**β** совпадали во всем, кроме своих номеров, то программа, образовавшаяся после поглощения команды **α** командой **β**, и первоначальная программа будут «работать совершенно одинаково».

Слова, взятые в предыдущей фразе в кавычки, уточняются следующим образом. Возьмем два экземпляра машины Поста, приведем каждый из них в некоторое – одно и то же для обеих машин – начальное состояние и заставим работать первую машину по некоторой программе А, а вторую – по некоторой программе Б. Предполагая, что машины работают синхронно, будем сравнивать одновременно возникающие состояния первой и второй машины. В начальный момент эти состояния

[1]Если бы мы хотели поглотить команду № 5 командой № 10, то надо было бы заменить отсылку 5 на отсылку 10 во всех командах, где она встречается (в данном случае в команде № 4), затем вычеркнуть команду № 5, после чего уменьшить на 1 все номера команд, следующих за вычеркиваемой, и все отсылки, совпадающие с этими номерами.

совпадают по условию. Может случиться, что они совпадают и во все последующие моменты, причем остановка каждой машины происходит – если только происходит вообще – одновременно с остановкой другой машины и имеет то же качество (т. е. эта остановка либо у обеих машин результативная, либо у обеих машин безрезультатная). Если описанное явление осуществляется для любого начального состояния, общего для обеих машин, то мы и будем говорить, что программы А и Б работают совершенно одинаково. Более грубо, А и Б работают совершенно одинаково, коль скоро для любого ***m***по прошествии ***m*** шагов работы программы А возникает то же состояние, что после ***m*** шагов работы программы Б, – при условии, что это было верно для ***m*=0**, т. е. в самом начале работы.

Посмотрим теперь, нельзя ли сократить и программу IV^8. В ней нет двух команд, различающихся только номерами, поэтому способ поглощения здесь может, вообще говоря, привести к программе, которая не будет работать совершенно одинаково с исходной. Все же и программу IV^8 можно сократить способом поглощения.

С этой целью сравним команды № 8 и №2. Их нельзя объединить в одну, поскольку они имеют разные отсылки. Оказывается, тем не менее, что команда № 2 может, в известном смысле, взять на себя функции команды № 8 (и потому поглотить последнюю).

В самом деле, пусть на некотором этапе работы программы нам предстоит выполнять команду №8. Предположим, что секция, расположенная непосредственно левее той, которая обозревается в рассматриваемый момент кареткой, пуста. Тогда команда №8 сдвинет каретку в эту пустую секцию, после чего команда №4 поставит на ней метку. Если вместо команды №8 мы станем выполнять команду №2, то произойдет вот что: команда №2 сдвинет каретку в ту же, что и раньше, пустую секцию, команда №3 заставит машину сделать «шаг на месте», после чего опять-таки сработает команда №4. Посмотрим, что случится, если секция, расположенная непосредственно левее обозреваемой, не пуста, а отмечена. Если так, то в первом случае (т.е. после выполнения команды №8) произойдет безрезультатная остановка, а во втором случае (т.е. после выполнения команды №2) таковой не произойдет.

Проведенное рассуждение показывает, что если последовательное выполнение команд №8 и №4 не приводит к безрезультатной остановке, то оно может быть заменено выполнением команд №№2, 3, 4 (в то время как обратная замена может, вообще говоря, существенно изменить работу программы: выполнение команд №№ 2,3,4 никогда не приводит к безрезультатной остановке, но при замене этого выполнения выполнением команд №№8, 4 может возникнуть безрезультатная остановка). Поэтому, если ограничиться такими начальными состояниями, для которых при выполнении программы IV^8 невозможна безрезультатная остановка (а к таким заведомо относятся все состояния из $\boldsymbol{D_n}$, где ***n***= 0, 1,2…), то программа IV^7, получающаяся после поглощения команды № 8 командой № 2, также будет решением задачи 4. Вот эта программа:

Программа IV^7

1. ? → 6, → 2
2. ⇐ 3
3. ? → 4, → 2
4. V5
5. стоп
6. ⇒7
7. ? → 6, → 2

Упражнение. Будут ли программы IV8 и IV7 работать совершенно одинаково?

Произведя в программе IV^7 поглощение команды №7 командой №1, получим следующую программу:

Программа IV^6

1. ? → 6, → 2

2. ⇐ 3
3. ? → 4, → 2
4. V5
5. стоп
6. ⇒1

Поскольку программы IV^7 и IV^6 работают совершенно одинаково, то программа IV^6, так же как и IV^7, будет решением задачи 4. Попробуйте доказать, что задача 4 не имеет более коротких решений.

Совершенно аналогично решается задача о прибавлении единицы для начальных состояний, в которых обозреваемая секция либо отмечена, либо стоит правее отмеченного массива. Решение соответствующей задачи – назовем ее задачей 4' – можно получить, заменив в произвольном решении задачи 4 все знаки ⇒ на ⇐, а знаки ⇐на⇒.

Если обозначить через $\boldsymbol{D'_n}$ совокупность всех такихсостояний из $\boldsymbol{E_n}$, в которых каретка стоит либо на массиве, либо правее массива, то можно заметить, что – при каждом $\boldsymbol{n}$– класс $\boldsymbol{E_n}$ есть объединение классов $\boldsymbol{D_n}$и $\boldsymbol{D'_n}$, а класс $\boldsymbol{B_n}$– пересечение этих классов:

$$D_n \cap D'_n = B_n$$
$$D_n \cup D'_n = E_n$$

Выполнение задания №4.

Задания на практическую работу №9

1. Машина Поста задается начальным состоянием, представленным на рисунке 3.10. Указать состояние машины после выполнения программы 1 и программы 2.

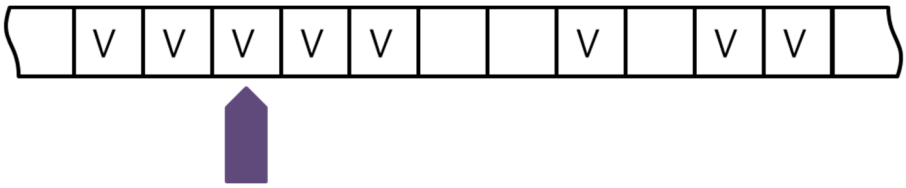


Рисунок 3.10 Начальное состояние машины Поста

Программа 1	Программа 2
1. ⇒3	1. ⇒3
2. ? → 7, → 11	2. ? → 7, → 11
3. ξ 4	3. ξ 4
4. ⇐ 2	4. ⇒11
5. ⇒ 6	5. V 5
6. ⇒ 10	6. ξ 15
7. V 5	7. ⇐ 8
8. стоп	8. ξ 15
9. ξ 12	9. ξ 12
10. ? → 6, → 14	10. ? → 6, → 14
11. ⇒13	11. ⇒13
12. ⇒11	12. ξ 4
13. V 5	13. ⇐ 2
14. ξ 15	14. ⇒ 6
15. ⇐ 8	15. стоп

2. Машина Поста задается начальным состоянием, представленным на рисунке 3.11. Написать программу заполняющая ячейки -1, 1 3, и стирающая ячейки -3 и 5.

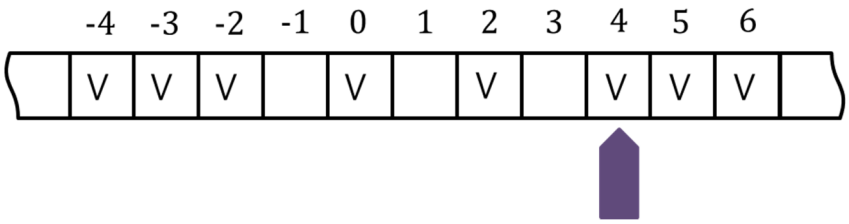


Рисунок 3.11 Начальное состояние машины Поста

3. Машина Поста задается начальным состоянием, представленным на рисунке 3.12. Написать программу заполняющая все пустые ячейки между двумя метками.

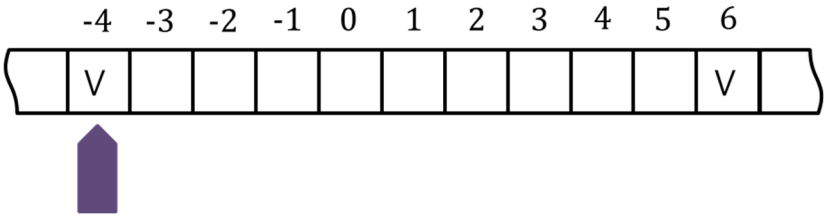


Рисунок 3.12 Начальное состояние машины Поста

4. Машина Поста задается начальным состоянием, представленным на рисунке 3.13а-в. Написать программу, выполняющую сложение текущего состояния с числом 3.

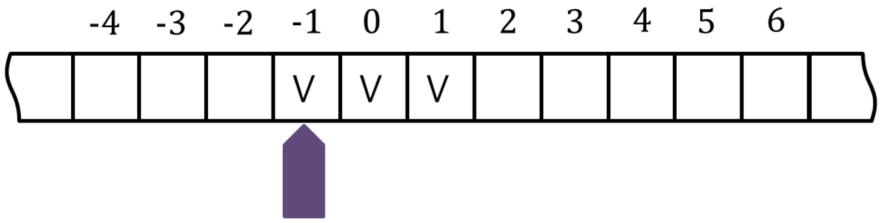


Рисунок 3.13а Начальное стояние машины Поста

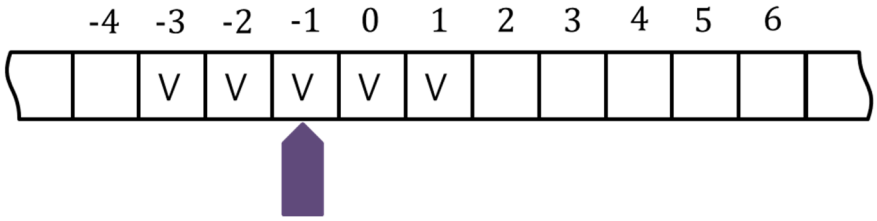


Рисунок 3.13б Начальное стояние машины Поста

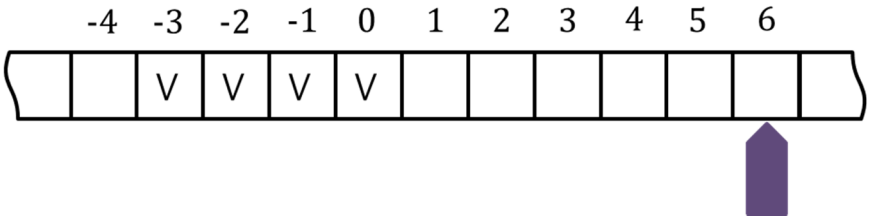


Рисунок 3.13в Начальное стояние машины Поста

Лабораторная работа №4

Создание и расчет локальной компьютерной сети

Цели работы.

1. Смоделировать локальную вычислительную сеть учебной аудитории.
2. Приобретение по разработке локальных вычислительных сетей.

Постановка задачи

1. Выбрать архитектуру сети.
2. Рассчитать длины кабелей и количество оборудования.
3. Рассчитать примерную стоимость сети

Содержание отчета

1. Структура сети.
2. Описание оборудования и характеристик.

Теоретические сведения.

Вычислительная сеть(ВС)состоит из вычислительных машин и сети передачи данных (сети связи). ВС классифицируются по геометрическим масштабам:

- ✓ глобальные вычислительные сети (англ. GlobalAreaNetwork, GAN);
- ✓ широкомасштабныевычислительные сети (англ. WideAreaNetwork, WAN);
- ✓ региональныевычислительные сети (англ. MetropolitanAreaNetwork, MAN);
- ✓ локальные вычислительные сети (англ. LocalAreaNetwork, LAN).

Под локальной вычислительной сетью (ЛВС) обычно понимают ВС, соединяющие вычислительные машины в одной комнате, здании или в нескольких близко расположенных зданиях. Сети связи ЛВС имеют в настоящее время следующие типичные характеристики:

- ✓ высокую скорость передачи данных (10^8–10^9 бит/с);
- ✓ небольшую протяженность (0.1-50 км);
- ✓ малую вероятность ошибки передачи данных (10^{-8} - 10^{-11}).

ЛВС - это система, составленная из отдельных модулей, которые можно добавлять и выстраивать в нужной конфигурации. Основными составными частями сети являются:

- ✓ абонентские станции;
- ✓ серверы сети;
- ✓ сетевые адаптеры;
- ✓ линии связи;
- ✓ терминаторы;
- ✓ ретрансляторы;
- ✓ сетевое программное обеспечение.

Кроме основных компонент, сеть может включать в состав блоки бесперебойного питания, резервные приборы, современные динамически распределяемые объекты и различные типы серверов (такие как файл-серверы, принт-серверы или архивные серверы).

Создавая ЛВС, разработчик стоит перед проблемой: при известных данных о назначении, перечне функций ЛВС и основных требованиях к комплексу технических и программных средств ЛВС построить сеть, то есть решить следующие задачи:

- ✓ определить архитектуру ЛВС: выбрать типы компонент ЛВС;
- ✓ рассчитать количество компонент ЛВС;
- ✓ произвести оценку показателей эффективности ЛВС;
- ✓ определить стоимость ЛВС.

При этом должны учитываться правила соединения компонентов ЛВС, основанные на стандартизации сетей, и их ограничения, специфицированные изготовителями компонент ЛВС.

Конфигурация ЛВС для РТ существенным образом зависит от особенностей конкретной прикладной области. Эти особенности сводятся к типам передаваемой информации (данные, речь,

графика), пространственному расположению абонентских систем, интенсивностям потоков информации, допустимым задержкам информации при передаче между источниками и получателями, объемам обработки данных в источниках и потребителях, характеристикам абонентских станций, внешним климатическим, электромагнитным факторам, эргономическим требованиям, требованиям к надежности, стоимости ЛВС и т.д.

Исходные данные для проектирования ЛВС могут быть получены в ходе предпроектного анализа прикладной области, для которой должна быть создана РТ. Эти данные уточняются затем п результате принятия решений на этапах проектирования ЛВС и построения все более точных моделей РТ, что позволяет "в техническом задании на ЛВС сформулировать требования к ней. Лучшая ЛВС - это та, которая удовлетворяет всем требованиям пользователей, сформулированным в техническом задании на разработку ЛВС, при минимальном объеме капитальных и эксплуатационных затрат. Данная лабораторная работа посвящена практическому изучению методов проектирования конфигурации ЛВС.

Проектирование ЛВС.

Проектирование конфигурации ЛВС относится к этапу проектирования технического обеспечения автоматизированных систем и осуществляется на этом этапе после распределения функции автоматизированной системы по абонентским станциям ЛВС, выбора типов абонентских станций, определения физического расположения абонентских станций.

Задание на проектирование включает требования к ЛВС, указания о доступных компонентах аппаратных и программных средств, знания о методах синтеза и анализа ЛВС, предпочтения и критерии сравнения вариантов конфигурации ЛВС.

Рассмотрим варианты топологии и состав компонент локальной вычислительной сети.

Топология ЛВС.

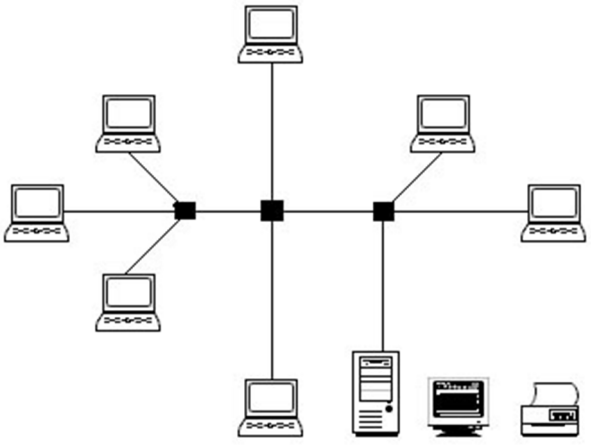
Топология сети определяется способом соединения ее узлов каналами связи. На практике используются 4 базовые топологии:

- ✓ звездообразная (рис.4.1);
- ✓ кольцевая (рис. 4.2);
- ✓ шинная (рис. 4.3);
- ✓ древовидная или иерархическая (рис. 4).

Топология сети влияет на надежность, гибкость, пропускную способность, стоимость сети и время ответа [1, табл. 1].

Выбранная топология сети должна соответствовать географическому расположению сети ЛВС, требованиям, установленным для характеристик сети, перечисленным в табл. 1. Топология влияет на длину линий связи.

Топология звезда



Топология распределенная звезда

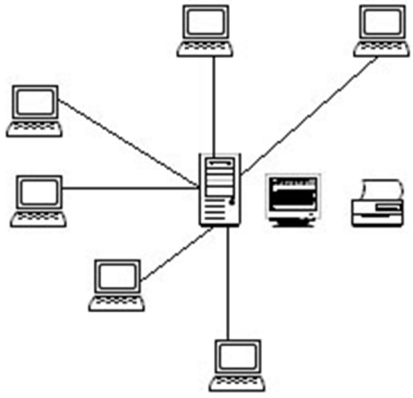


Рисунок 4.1. Звездообразная топология

Таблица 1.
Сравнительные данные по характеристикам ЛВС

Характеристика	Качественная оценка характеристик		
	Шинной и древовидной сети	**Кольцевой сети**	**Звездообразной сети**
1. Время ответа	В маркерной шине $t_{отв.}$ предсказуемо и зависит от числа узлов сети. В случайной шине $t_{отв.}$ зависит от нагрузки	$t_{отв.}$ есть функция от числа узлов сети	$T_{отв.}$ зависит от нагрузки и временных характеристик центрального узла
2. Пропускная способность С	В маркерной шине зависит от количества узлов. В случайной шине С увеличивается при спорадических малых нагрузках и падает при обмене длинными сообщениями в стационарном режиме	С падает при добавлении новых узлов	С зависит от производительности центрального узла и пропускной способности абонентских каналов
3. Надежность	Отказы АС не влияют на работоспособность остальной части сети. Разрыв кабеля выводит из строя шинную ЛВС.	Отказ одной АС не приводит к отказу всей сети. Однако использование обходных схем позволяет защитить сеть от отказов АС	Отказы АС не влияют на работоспособность остальной части сети. Надежность ЛВС определяется надежностью центрального узла

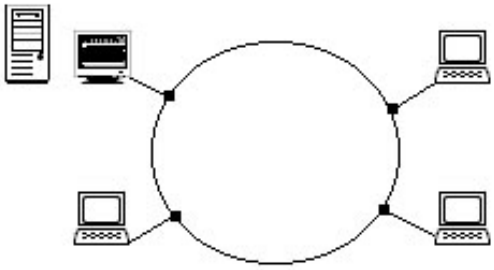


Рисунок 4.2 Топология кольцо

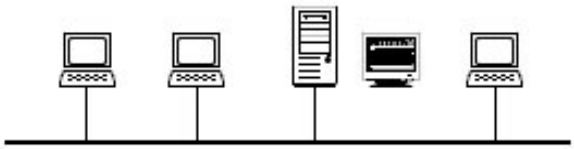


Рисунок 4.3 Топология линейная шина

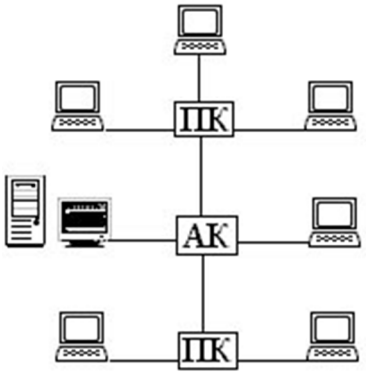


Рисунок 4.4 Иерархическая сеть с концентраторами.
АК – активный концентратор ПК – пассивный концентратор

Выбор типов линий связи

В качестве линий связи могут выступать кабели со скрученными парами проводов *(витые пары), коаксиальные кабели, волоконно-оптические кабели, радио, инфракрасные* ИК-, СВЧ - каналы.

В набор параметров линий связи ЛВС входят: полоса пропускания и скорость передачи данных, способность к двухточечной, многоточечной и/или широковещательной передаче (то есть допустимые применения), максимальная протяженность и число подключаемых абонентских систем, топологическая гибкость и трудоемкость прокладки, устойчивость к помехам и стоимость.

При выборе типов кабеля учитывают следующие показатели:

- стоимость монтажа и обслуживания;
- скорость передачи информации;
- ограничения на величину расстояния передачи информации (без дополнительных усилителей-повторителей *(repeater);*
- безопасность передачи данных.

Главная проблема заключается в одновременном обеспечении показателей, например, наивысшая скорость передачи данных ограничена максимально возможным расстоянием передачи данных, при котором еще обеспечивается требуемый уровень защиты данных. Легкая наращиваемость и простота расширения кабельной системы влияют на ее стоимость.

Условия физического расположения помогают определить наилучшим образом тип кабеля и его топологию. Каждый тип кабеля имеет собственные ограничения по максимальной длине:витая пара обеспечивает работу на коротких отрезках,одноканальный коаксиальный кабель - на больших расстояниях,многоканальный коаксиальный а волоконно-оптический кабель - на очень больших расстояниях.

Скорость передачи данных тоже ограничена возможностями кабеля: самая большая - уволоконно-оптического, затем идутодноканальный коаксиальный, многоканальный кабели и витая пара. Под требуемые характеристики можно подобрать имеющиеся в наличии кабели.

В табл. 2 приводятся характеристики линий связи ЛВС Ethernet.

Таблица 2.

Характеристики линий связи Ethernet

Параметр	**Тип линии связи**		
	Тонкопроводная (моноканал)	**Толстопроводная (моноканал)**	**Широкополосная (поликанал)**
Максимальная длина (без повторителен),м	185	500	1900
Тип кабеля	RG58	Коаксиальный кабель в тефлоновой или полихлорвиниловой оболочке	Телевизионный коаксиальный кабель
Максимальное число АС	30	200	1023
Скорость передачи, Мбит/с	10	10	10

Витая пара

Наиболее дешевым кабельным соединением является двухжильное соединение витым проводом, часто называемое витой парой" (twistedpair). Она позволяет передавать информацию со скоростью до 10 Мбит/с, легко наращивается, однако является помехонезащищенной. Длина кабеля не может превышать 1000 м при скорости передачи 1 Мбит/с. Преимуществами являются низкая цена и беспроблемная установка. Для повышения помехозащищенности информации часто используют экранированную витую пару. Это увеличивает стоимость витой пары и приближает ее цену к цене коаксиального кабеля.

Коаксиальный кабель

Коаксиальный кабель имеет среднюю цену, помехозащищен и применяется для связи на большие расстояния (несколько километров). Скорость передачи информации от 1 до 10 Мбит/с, а в некоторых

случаях может достигать 50 Мбит/с.Коаксиальный кабель используется для основной и широкополосной передачи информации.

Широкополосный коаксиальный кабель

Широкополосный коаксиальный кабель невосприимчив к помехам, легко наращивается, но цена его высокая. Скорость передачи информации равна 500 Мбит/с. При передаче информации в базисной полосе частот на расстояние более 1,5 км требуется усилитель, или так называемый повторитель (repeater). Поэтому суммарное расстояние при передаче информации увеличивается до 10 км. Для вычислительных сетей с топологией шина или дерево коаксиальный кабель должен иметь на конце согласующий резистор (terminator).

Ethernet-кабель

Ethernet-кабель также является коаксиальным кабелем с волновым сопротивлением 50 Ом. Его называют ещетолстыйEthernet(thick) или желтый кабель (yellowcable). Он использует 15-контактное стандартное включение. Вследствие помехозащищенности является дорогой альтернативой обычным коаксиальным кабелям. Максимально доступное расстояние без повторителя не превышает 500 м, а общее расстояние сети Ethernet -около 3000 м. Ethernet-кабель, благодаря своей магистральной топологии, использует в конце лишь один нагрузочный резистор.

Cheapernet-кабель

Более дешевым, чем Ethernet-кабель, является соединение Cheapernet-кабель или, как его часто называют,тонкий (thin)Ethernet. Это также 50-омный коаксиальный кабель со скоростью передачи информации в 10 миллионов бит/с.

При соединении сегментовCheapernet-кабеля также требуются повторители. Вычислительные сети cCheapernet-кабелем имеют небольшую стоимость и минимальные затраты при наращивании. Соединение сетевых плат производится с помощью широко используемых малогабаритных байонетных разъемов(СР-50). Дополнительного экранирования не требуется. Кабель присоединяется к ПК с помощью тройниковых соединителей (Tconnectors). Расстояние между двумя рабочими станциями без повторителей можетсоставлять максимум 300 м, а общее расстояние для сети наCheapemet-кабеле - около 1000 м. ПриемопередатчикCheapernetрасположен на сетевой плате и используется как для гальванической развязки между адаптерами, так и для усиления внешнего сигнала.

Оптоволоконные линии

Наиболее дорогими являютсяоптопроводники,называемые также стекловолоконным кабелем. Скорость распространения информации по ним достигает нескольких гигабит в секунду. Допустимое удаление более 50 км. Внешнее воздействие помех практически отсутствует. Применяются там, где возникают электромагнитные поля помех или требуется передача информации на очень большие расстояния без использования повторителей. Они обладают противоподслушивающими свойствами, так как техника ответвлений в оптоволоконных кабелях очень сложна. Оптопроводники объединяются в ЛВС с помощью звездообразного соединения.

Этапы конфигурирования ЛВС

Конфигурирование ЛВС - это многокритериальная оптимизационная задача, так как на выбор конфигурации ЛВС влияет большое число факторов. В качестве целевой функции при решении этой задачи можно взять минимизацию величины стоимости ее аппаратного и программного обеспечения при условиях удовлетворения всех требований пользователя к передаче информации в полном объеме, времени ответа, пропускной способности и надежности сети.

Проектирование конфигурации ЛВС требует решения ряда задач, включающих выбор комплекса программно-аппаратных средств локальной вычислительной сети, выбор типов сетей связи в данном комплексе, трассировку кабельной сети ЛВС в зданиях и помещениях. В процессе построения ЛВС необходимо учитывать ряд требований прикладного характера, например, физическое расположение пользователей, количество и типы оконечных систем, требования к передаче данных (типы данных, среднюю нагрузку), требования пользователей к программным и аппаратным ресурсам. Расстояние между оконечными системами, наличие несовместимых оконечных систем и требование к контролю доступа пользователей к отдельным участкам ЛВС могут привести к необходимости предусматривать в составе сети различныешлюзы и мосты. В любой ЛВС существенным фактором является максимально достижимая пропускная способность сети связи. Она характеризует предел допустимых функциональных возможностей сети. Поэтому перед выбором ЛВС необходимо оценить, какая пропускная способность требуется пользователям данной прикладной области.

Вот некоторые отличительные характеристики и факторы, влияющие на выбор комплекса программно-аппаратных средств локальных вычислительных сетей и проектирование соответствующей конфигурации:

1) характеристики среды передачи информации или кабельной системы, такие как: помехозащищенность, защита от климатических воздействий, протяженность без промежуточного усиления сигнала, стоимость приобретения и установки;

2) максимальная протяженность сети;

3) предполагаемое количество оконечных систем;

4) основная сфера применения (на производственном предприятии, в учреждении или в учебной сфере);

5) функциональное назначение, то есть классы решаемых задач (научная деятельность, образование, резервирование мест, удаленный ввод/вывод, "распределенная обработка данных, управление и учет, финансовые операции);

6) тип передаваемой информации (данные, изображения, речь);

7) оценка пропускной способности сети;

8) сетевое программное обеспечение;

9) интерсетевое обеспечение (необходима ли связь с другими сетями ЭВМ);

10) показатель надежности сети в целом и отдельных ее частей;

Проектирование конфигурации ЛВС проходит через три основных этапа:

1) определение требований к ЛВС;

2) синтез альтернативных конфигураций ЛВС;

3) выбор наиболее предпочтительной конфигурации из имеющихся вариантов.

Проектирование ЛВС необходимо производить с учетом стратегического планирования развития РТ, принимая во внимание возможность увеличения количества, АС в ЛВС, подключения новых участков ЛВС в других подразделениях предприятия (учреждения).

Исходные данные для проектирования ЛВС представляют собой формальное описание конкретной прикладной области (например, цеха механообработки, администрации производственного объединения, бухгалтерии, отдела кадров и т.д.). Основой является план зданий и помещений с отмеченными на нем местоположениями существующих ЭВМ.

Выполнение работы

Исходные данные к заданию

Пользователи: студенты, преподаватели, инженеры, программисты, лаборанты.

Функции:

1) реализация учебного процесса на лабораторных, практических занятиях, выполнение курсового и дипломного проектирования;

2) организация учебного процесса, подготовка к проведению занятий, разработка методического обеспечения;

3) разработка программного обеспечения для работы в сети;

4) профилактика и ремонт оборудования.

Требования к проектируемой сети

Проектируемая ЛВС должна удовлетворять целому ряду требовании. Наиболее значительные из них связаны с передачей данных и состоят в следующем:

ЛВС должна выполнять разнообразные функции по передаче данных, включая пересылку файлов, поддержку терминалов (в том числе графических), электронную почту, обмен с внешними запоминающими устройствами, обработку сообщений доступ к файлам и базам данных.

ЛВС должна допускать подключение большого набора стандартных и специальных устройств, в том числе: ЭВМ, терминалов, устройств внешней памяти, принтеров, графопостроителей, факсимильных устройств, контрольного и управляющего оборудования, аппаратуры подключения к другим ЛВС и сетям (в том числе и к телефонным) и т.д.

ЛВС должна доставлять данные адресату с высокой степенью надежности (коэффициент готовности сети должен быть не менее 0.96), должна соответствовать существующим стандартам, обеспечивать "прозрачный" режим передачи данных, допускать простое подключение новых устройств и отключение старых без нарушения работы сети длительностью не более 1 с ; достоверность передачи данных должна быть не больше +1E-8.

Перечень задач по проектированию ЛВС

- ✓ Выбрать топологию ЛВС (и обосновать выбор).
- ✓ Нарисовать функциональную схему ЛВС и составить перечень аппаратных средств.
- ✓ Выбрать оптимальную конфигурацию ЛВС.
- ✓ Произвести ориентировочную трассировку кабельной сети и выполнить расчет длины кабельного соединения для выбранной топологии с учетом переходов между этажами. Поскольку существуют ограничения на максимальную длину одного сегмента локальной сети для определенного типа кабеля и заданного количества рабочих станции, требуется установить необходимость использования повторителей.
- ✓ Рассчитать надежность и стоимость ЛВС.

Содержание отчета

Перечень этапов проектирования конфигурации ЛВС с указанием принятых проектных решений.
Функциональная схема ЛВС.
Результаты расчетов стоимости и надежности ЛВС.