О. Г. Иванова, Ю. В. Кулаков,
Н. Г. Шахов, В. Г. Однолько

ПРАКТИКУМ ПО ИНФОРМАТИКЕ

Тамбов
• Издательство ФГБОУ ВПО «ТГТУ» •
2014

Министерство образования и науки Российской Федерации

Федеральное государственное бюджетное образовательное учреждение
высшего профессионального образования
«Тамбовский государственный технический университет»

**О. Г. ИВАНОВА, Ю. В. КУЛАКОВ,
Н. Г. ШАХОВ, В. Г. ОДНОЛЬКО**

ПРАКТИКУМ ПО ИНФОРМАТИКЕ

Допущено Учебно-методическим объединением вузов
по университетскому политехническому образованию
в качестве учебного пособия для студентов высших учебных заведений,
обучающихся по направлениям подготовки
230400 «Информационные системы и технологии» и
220100 «Системный анализ и управление»

Тамбов
• Издательство ФГБОУ ВПО «ТГТУ» •
2014

УДК 004 (075.8)
ББК $з$81я73-5
И20

Р е ц е н з е н т ы:

Доктор физико-математических наук, профессор,
заслуженный деятель науки РФ
В. Ф. Крапивин

Доктор физико-математических наук, профессор
Ф. А. Мкртчян

И20 **Практикум** по информатике : учебное пособие / О. Г. Иванова, Ю. В. Кулаков, Н. Г. Шахов, В. Г. Однолько. – Тамбов : Изд-во ФГБОУ ВПО «ТГТУ», 2014. – 112 с. – 100 экз.
ISBN 978-5-8265-1349-1.

Содержит теоретический материал, лабораторные работы и список рекомендуемой литературы.

Предназначено для студентов высших учебных заведений, обучающихся по специальности 10.05.03 (090303) «Информационная безопасность автоматизированных систем» и направлениям 13.03.02 (140400) «Электроэнергетика и электротехника», 27.03.03 (220100) «Системный анализ и управление», 09.03.01 (230100) «Информатика и вычислительная техника», 09.03.02 (230400) «Информационные системы и технологии», и для студентов средних учебных заведений, обучающихся по специальности 09.02.03 (230115) «Программирование в компьютерных системах».

УДК 004 (075.8)
ББК $з$81я73-5

ISBN 978-5-8265-1349-1

ВВЕДЕНИЕ

В учебном пособии представлены семнадцать лабораторных работ, охватывающих практически все разделы современного базового курса информатики.

Лабораторная работа № 1 посвящена решению задач, связанных с определением количества информации.

В лабораторных работах № 2 – 4 рассматриваются задачи представления информации в цифровых автоматах.

Лабораторная работа № 5 посвящена логическим основам построения цифровых автоматов.

В лабораторных работах № 6 и 7 рассматриваются математические модели, предназначенные для строгого уточнения наивного определения понятия алгоритма.

Лабораторные работы № 8 – 15 посвящены основам алгоритмизации и программирования на языке высокого уровня.

В лабораторной работе № 16 рассматривается задача контроля правильности работы запоминающих устройств.

Лабораторная работа № 17 посвящена решению задачи криптоанализа одного из исторических шифров.

Каждая лабораторная работа содержит цель, задания в тридцати вариантах, основные теоретические положения и пример выполнения задания.

В конце пособия даётся список рекомендованной литературы.

Пособие соответствует содержанию и требованиям ФГОС ВПО по направлению подготовки 09.03.02 (230400) «Информационные системы и технологии» и может быть востребовано при освоении основной образовательной программы по родственному направлению 27.03.03 (220100) «Системный анализ и управление».

Несмотря на наличие ранее изданной учебной литературы, данное пособие представляет и методически объединяет все разделы базового курса информатики и призвано создать у учащихся целостное представление об этой науке.

Надеемся, что данное учебное пособие будет способствовать приобретению студентами необходимых знаний, умений и навыков, которые помогут им не только в изучении дисциплин общенаучного и профессионального циклов, но и в решении многих практических задач.

Лабораторная работа № 1

ИЗМЕРЕНИЕ ИНФОРМАЦИИ

Цель: знакомство с понятиями «информационный объём» и «количество информации», а также единицами измерения информации.

Задание.

1. Определить количество информации в битах в сообщении о цвете выбранного наугад шара из урны с n шарами, каждый из которых имеет свой индивидуальный цвет.

2. Определить количество информации в битах в сообщениях о случайном выборе из урны с g жёлтыми, z зелёными, k красными и s синими шарами шара каждого цвета. Ответы (при необходимости) округлить до сотых.

3. Определить среднее значение количества информации в битах в сообщении о цвете выбранного наугад шара из урны с g жёлтыми, z зелёными, k красными и s синими шарами. Ответ округлить до сотых.

4. Определить информационный объём в байтах текста на a страницах, если на каждой странице имеется b строк по c символов в строке. Известно, что при создании этого текста использован алфавит мощности d.

5. Определить информационный объём в килобайтах изображения размером $l \times m$ точек, если каждая точка этого изображения может иметь один из r цветов.

Значения n, g, z, k, s, a, b, c, d, l, m, r приведены в табл. 1.1.

Основные положения

Информация (от лат. *informatio* – разъяснение, изложение) – это первичное понятие информатики и поэтому не имеет строгого определения. Первоначально под информацией понимались сведения, передаваемые людьми устным, письменным или каким-либо другим способом [1], а с середины XX в. понятие информации стало общенаучным понятием, включающим в себя обмен сведениями между людьми, человеком и автоматом, автоматом и автоматом.

При реализации информационных процессов всегда происходит перенос информации в пространстве и времени от источника информации к её получателю в виде *сообщений* – последовательностей знаков (символов) естественного или формального языка. Сообщение может изучаться на трёх уровнях: *синтаксическом*, где рассматриваются его внутренние свойства; *семантическом*, когда анализируется его смысловое содержание; и *прагматическом*, где рассматривается его потребительские качества.

Для измерения информации на синтаксическом уровне, когда оперируют с обезличенной информацией о разных по своей природе объектах, существуют два подхода: *энтропийный* и *объёмный*.

Таблица 1.1

Вариант	n	g	z	k	s	a
1	64	8	16	4	4	4
2	512	21	2	20	21	6
3	2048	16	8	4	4	6
4	128	18	4	7	3	2
5	8	20	4	5	3	5
6	4	17	8	4	3	6
7	32	20	4	4	4	2
8	1024	21	8	14	21	6
9	2048	39	4	12	9	4
10	32	16	8	4	4	3
11	256	13	2	5	12	6
12	8	17	1	6	8	6
13	64	10	4	9	9	6
14	2048	34	2	20	8	2
15	128	19	2	7	4	3
16	256	9	16	4	3	6
17	1024	12	4	14	2	4
18	2048	20	2	6	4	3
19	512	16	4	4	8	5
20	32	19	1	6	6	6
21	4	26	1	12	25	3
22	128	10	8	4	10	6
23	8	15	8	4	5	2
24	256	10	2	10	10	5
25	32	29	8	18	9	6
26	2048	7	16	3	6	5
27	1024	26	8	18	12	3
28	2048	6	16	8	2	3
29	8	25	4	30	5	4
30	256	10	16	5	1	2

Продолжение табл. 1.1

Вариант	*b*	*c*	*d*	*l*	*m*	*r*
1	20	56	4	4096	2048	256
2	20	40	8	4096	2048	128
3	21	56	8	1024	512	16
4	20	40	256	512	256	64
5	24	32	256	2048	1024	4
6	27	64	1024	1024	512	128
7	20	48	128	512	256	128
8	28	64	16	1024	512	128
9	23	40	8	2048	1024	64
10	22	48	1024	1024	512	64
11	25	32	8	512	256	8
12	30	56	16	1024	512	4
13	28	56	512	512	256	64
14	20	40	8	512	256	8
15	30	56	16	4096	2048	128
16	28	32	1024	2048	1024	16
17	21	48	4	4096	2048	256
18	21	56	32	512	256	8
19	29	56	16	512	256	64
20	27	56	64	512	256	32
21	27	40	512	2048	1024	32
22	22	56	1024	2048	1024	8
23	20	40	4	2048	1024	64
24	30	48	512	4096	2048	16
25	29	40	64	1024	512	128
26	28	40	32	2048	1024	128
27	24	48	512	4096	2048	64
28	24	56	8	512	256	32
29	23	40	4	1024	512	4
30	25	48	32	1024	512	256

При энтропийном подходе, принятом в теории информации и кодирования, говорят о *количестве информации* в сообщении. Количество информации I в сообщении определяется как мера уменьшения энтропии (неопределённости) состояния данной системы после получения этого сообщения: $I = H_{apr} - H_{aps}$, где H_{apr}, H_{aps} – априорная и апостериорная энтропия состояния исследуемой системы соответственно. В случае когда сообщение снимает неопределённость полностью ($H_{aps} = 0$), количество получаемой информации совпадает с первоначальной энтропией, т.е. количество информации $I = H_{apr}$.

Если множество событий состоит из n равновероятных событий, то в качестве меры неопределённости может быть принята величина логарифма от числа событий $H = \log n$. Эта мера была предложена в 1928 г. американским учёным Р. Хартли (1888 – 1970). При этом основание логарифма может быть различным, но поскольку современная информационная техника базируется на элементах, имеющих два устойчивых состояния, в качестве основания используют число 2, а единицу неопределённости называют *битом* (англ. *bit*, от *binary* – двоичный и *digit* – знак).

Таким образом, количество информации I в сообщении о наступлении одного из n равновероятных событий определяется по формуле $I = \log_2 n$, бит. Другими словами, количество информации равно степени, в которую необходимо возвести число 2, чтобы получить n, или сколько вопросов, предполагающих ответы «да» и «нет», необходимо и достаточно задать, чтобы однозначно определить наступившее событие.

Американский инженер и математик К. Шеннон (1916 – 2001) обобщил понятие меры неопределённости на случай неравновероятных событий. Если некоторое j-е событие из n событий может наступить с вероятностью p_j, то количество информации в сообщении о наступлении этого события определяется по (первой) формуле $I_j = \log_2 (1/p_j)$, бит, а среднее значение количества информации, приходящееся на одно событие – по (второй) формуле $I = \sum_{j=1}^{n} p_j \log_2 (1/p_j)$, бит.

При объёмном подходе к измерению информации говорят об *информационном объёме* или *информационной ёмкости* сообщения. Здесь сообщение рассматривается как совокупность символов какого-либо алфавита, каждый символ закодирован (при двоичном кодировании) последовательностью бит, а информационная ёмкость сообщения определяется как произведение числа символов и информационного объёма одного символа. При этом информационный объём символа определяется по формуле $I = \log_2 n$, бит, где n – *мощность алфавита* (общее число символов в алфавите).

Кроме единицы измерения информации бит, используют *байт* (1 байт = 8 бит), *килобайт* (1 Кбайт = 1024 байт), *мегабайт* (1 Мбайт = = 1024 Кбайт), *гигабайт* (1 Гбайт = 1024 Мбайт), *терабайт* (1 Тбайт = = 1024 Гбайт) и более крупные единицы измерения информации.

Пример выполнения задания

1. Определить количество информации в битах в сообщении о цвете выбранного наугад шара из урны с n шарами, каждый из которых имеет свой индивидуальный цвет.

Дано: $n = 16$.

Решение.

Поскольку шар каждого цвета может быть выбран с равной вероятностью, применим формулу Хартли:

$I = \log_2 n = \log_2 16 = 4$ бита.

Ответ: 4 бита.

2. Определить количество информации в битах в сообщениях о случайном выборе из урны с g жёлтыми, z зелёными, k красными и s синими шарами шара каждого цвета. Ответы (при необходимости) округлить до сотых.

Дано: $g = 16$, $z = 2$, $k = 10$, $s = 4$.

Решение.

Заметим, что события, заключающиеся в выборе шаров разного цвета, неравновероятны.

Вычислим вероятности выбора шара жёлтого, зелёного, красного и синего цветов:

$p_g = g / (g + z + k + s) = 16 / (16 + 2 + 10 + 4) = 16 / 32 = 0{,}5$;

$p_z = z / (g + z + k + s) = 2 / 32 = 0{,}0625$;

$p_k = k / (g + z + k + s) = 10 / 32 = 0{,}3125$;

$p_s = s / (g + z + k + s) = 4 / 32 = 0{,}125$.

Количество информации в сообщениях о выборе шара каждого цвета вычислим с использованием (первой) формулы Шеннона:

$I_g = \log_2 (1 / p_g) = \log_2 (1 / 0{,}5) = \log_2 (2) = 1$ бит;

$I_z = \log_2 (1 / p_z) = \log_2 (1 / 0{,}0625) = \log_2 (16) = 4$ бита;

$$I_k = \log_2 (1 / p_k) = \log_2 (1 / 0{,}3125) = \log_2 (3{,}2) = \frac{\ln 3{,}2}{\ln 2} = \frac{1{,}163151}{0{,}693147} =$$

$= 1{,}678073 \approx 1{,}68$ бита;

$I_s = \log_2 (1 / p_s) = \log_2 (1 / 0{,}125) = \log_2 (8) = 3$ бита.

Ответ: 1 бит; 4 бита; 1,68 бита; 3 бита.

3. Определить среднее значение количества информации в битах в сообщении о цвете выбранного наугад шара из урны с g жёлтыми, z зелёными, k красными и s синими шарами. Ответ округлить до сотых.

Дано: $g = 16$, $z = 2$, $k = 10$, $s = 4$.

Решение.

Полученные в предыдущем примере для тех же исходных данных вероятности выбора жёлтого, зелёного, красного и синего цветов обозначим переменными с числовыми индексами:

$$p_1 = p_g = 0{,}5; \quad p_2 = p_z = 0{,}0625; \quad p_3 = p_k = 0{,}3125; \quad p_4 = p_s = 0{,}125.$$

Среднее значение количества информации, приходящееся на один из четырёх цветов, вычислим по (второй) формуле Шеннона:

$$I = \sum_{j=1}^{4} p_j \log_2 (1/p_j) =$$

$$= p_1 \log_2 (1/p_1) + p_2 \log_2 (1/p_2) + p_3 \log_2 (1/p_3) + p_4 \log_2 (1/p_4) =$$

$$= 0{,}5 \cdot 1 + 0{,}0625 \cdot 4 + 0{,}3125 \cdot 1{,}678073 + 0{,}125 \cdot 3 = 1{,}64940 \approx 1{,}65 \text{ бита}.$$

Ответ: 1,65 бита.

4. Определить информационный объём в байтах текста на a страницах, если на каждой странице имеется b строк по c символов в строке. Известно, что при создании этого текста использован алфавит мощности d.

Дано: $a = 3$, $b = 24$, $c = 48$, $d = 32$.

Решение.

Определим информационный объём одного символа текста:

$I_c = \log_2 d = \log_2 32 = 5$ бит.

Информационный объём всего текста

$I = abcI_c = 3 \cdot 24 \cdot 48 \cdot 5 = 17\,280$ бит $= 17\,280$ бит : 8 бит / байт =

$= 2160$ байт.

Ответ: 2160 байт.

5. Определить информационный объём в килобайтах изображения размера $l \times m$ точек, если каждая точка этого изображения может иметь один из r цветов.

Дано: $l = 512$, $m = 256$, $r = 16$.

Решение.

Определим информационный объём одной точки изображения $I_т = \log_2 r = \log_2 16 = 4$ бита.

Информационный объём изображения

$I = lmI_т = 512 \cdot 256 \cdot 4 = 524\,288$ бит = 524 288 бит : 8 бит / байт =

= 65 536 байт = 65 536 байт : 1024 байт / Кбайт = 64 Кбайта.

Ответ: 64 Кбайта.

Лабораторная работа № 2

ПРЕДСТАВЛЕНИЕ ЧИСЕЛ В ПОЗИЦИОННЫХ СИСТЕМАХ СЧИСЛЕНИЯ

Цель: знакомство с понятием позиционной системы счисления и приобретение навыков перевода чисел из одной позиционной системы счисления в другую.

Задание. Выполнить пять переводов чисел, заданных в табл. 2.1.

Основные положения

Системой счисления (СС) называется совокупность приёмов наименования и записи чисел [2]. СС называется *позиционной*, если вес каждой цифры зависит от её положения в последовательности цифр, изображающих число. *Основанием K* позиционной СС называется число единиц какого-либо разряда, заменяемых единицей старшего разряда. Позиционная СС с основанием *K* называется *K-ичной* СС. Для записи чисел в *K*-ичной СС используются *K* цифр, обозначающих числа 0, 1, …, $K - 1$. Так, в табл. 2.2 представлены первые шестнадцать целых неотрицательных чисел в позиционных системах счисления с некоторыми основаниями.

Под *переводом* числа из *P*-ичной системы счисления в *Q*-ичную понимается преобразование исходного *P*-ичного представления числа в представление этого же числа в *Q*-ичной СС. Такой перевод будем обозначать схематично в виде $P \to Q$.

Рассмотрим различные варианты перевода чисел из одной системы счисления в другую. При переводе $P \to 10$ (из некоторой *P*-ичной системы счисления в 10-ичную) сначала исходную запись числа представляют в виде полинома. Затем в полиноме все *P*-ичные представления чисел заменяют 10-ичными и вычисляют значение этого полинома средствами десятичной арифметики.

Таблица 2.1

Вариант	Задание		
	1	2	3
1	$10011{,}1011_2 = \_\_{}_{10}$	$25{,}8125_{10} = \_\_{}_2$	$3131232{,}3202_4 = \_\_{}_2$
2	$1230{,}21_4 = \_\_{}_{10}$	$108{,}5625_{10} = \_\_{}_4$	$33556{,}704_8 = \_\_{}_2$
3	$142{,}13_5 = \_\_{}_{10}$	$39{,}84_{10} = \_\_{}_5$	$376e{,}e2_{16} = \_\_{}_2$
4	$245{,}3_6 = \_\_{}_{10}$	$203{,}5_{10} = \_\_{}_6$	$603f2{,}564_{16} = \_\_{}_4$
5	$137{,}26_8 = \_\_{}_{10}$	$158{,}1875_{10} = \_\_{}_8$	$12301213{,}231_4 = \_\_{}_2$
6	$1ea{,}c_{16} = \_\_{}_{10}$	$379{,}5_{10} = \_\_{}_{16}$	$66147{,}55_8 = \_\_{}_2$
7	$11001{,}1101_2 = \_\_{}_{10}$	$22{,}25_{10} = \_\_{}_2$	$6c67{,}b4_{16} = \_\_{}_2$
8	$232{,}12_4 = \_\_{}_{10}$	$57{,}8125_{10} = \_\_{}_4$	$2d55a7{,}9ed_{16} = \_\_{}_4$
9	$234{,}14_5 = \_\_{}_{10}$	$47{,}32_{10} = \_\_{}_5$	$3333000{,}132_4 = \_\_{}_2$
10	$153{,}3_6 = \_\_{}_{10}$	$69{,}5_{10} = \_\_{}_6$	$37700{,}36_8 = \_\_{}_2$
11	$215{,}62_8 = \_\_{}_{10}$	$141{,}78125_{10} = \_\_{}_8$	$3fc0{,}78_{16} = \_\_{}_2$
12	$17b{,}8_{16} = \_\_{}_{10}$	$725{,}625_{10} = \_\_{}_{16}$	$16abf5{,}5ac_{16} = \_\_{}_4$
13	$10110{,}01_2 = \_\_{}_{10}$	$15{,}625_{10} = \_\_{}_2$	$1003320{,}321_4 = \_\_{}_2$
14	$321{,}31_4 = \_\_{}_{10}$	$58{,}4375_{10} = \_\_{}_4$	$10370{,}71_8 = \_\_{}_2$
15	$124{,}41_5 = \_\_{}_{10}$	$195{,}68_{10} = \_\_{}_5$	$10f8{,}e4_{16} = \_\_{}_2$
16	$535{,}3_6 = \_\_{}_{10}$	$77{,}5_{10} = \_\_{}_6$	$1b6ef9{,}be_{16} = \_\_{}_4$
17	$236{,}14_8 = \_\_{}_{10}$	$102{,}875_{10} = \_\_{}_8$	$1132222{,}1112_4 = \_\_{}_2$
18	$2d5{,}a_{16} = \_\_{}_{10}$	$495{,}3125_{10} = \_\_{}_{16}$	$13652{,}254_8 = \_\_{}_2$
19	$1111{,}101_2 = \_\_{}_{10}$	$28{,}375_{10} = \_\_{}_2$	$17aa{,}56_{16} = \_\_{}_2$
20	$322{,}13_4 = \_\_{}_{10}$	$103{,}125_{10} = \_\_{}_4$	$556aa{,}ffc_{16} = \_\_{}_4$
21	$1240{,}32_5 = \_\_{}_{10}$	$148{,}44_{10} = \_\_{}_5$	$110133{,}311_4 = \_\_{}_2$
22	$314{,}3_6 = \_\_{}_{10}$	$118{,}5_{10} = \_\_{}_6$	$7525{,}52_8 = \_\_{}_2$
23	$237{,}4_8 = \_\_{}_{10}$	$159{,}5_{10} = \_\_{}_8$	$1223113{,}2322_4 = \_\_{}_2$
24	$1a8{,}f_{16} = \_\_{}_{10}$	$424{,}9375_{10} = \_\_{}_{16}$	$15327{,}564_8 = \_\_{}_2$
25	$11100{,}011_2 = \_\_{}_{10}$	$19{,}6875_{10} = \_\_{}_2$	$1ad7{,}ba_{16} = \_\_{}_2$
26	$1213{,}02_4 = \_\_{}_{10}$	$46{,}375_{10} = \_\_{}_4$	$1be48{,}63_{16} = \_\_{}_4$
27	$1043{,}21_5 = \_\_{}_{10}$	$69{,}36_{10} = \_\_{}_5$	$1320013{,}333_4 = \_\_{}_2$
28	$205{,}3_6 = \_\_{}_{10}$	$101{,}5_{10} = \_\_{}_6$	$17007{,}77_8 = \_\_{}_2$
29	$146{,}7_8 = \_\_{}_{10}$	$95{,}34375_{10} = \_\_{}_8$	$1e07{,}fc_{16} = \_\_{}_2$
30	$1ef{,}5_{16} = \_\_{}_{10}$	$490{,}75_{10} = \_\_{}_{16}$	$100af5{,}6f8_{16} = \_\_{}_4$

Продолжение табл. 2.1

Вариант	Задание	
	4	5
1	$11010110101111,1011101_2 = \_\_\_4$	$1223113,2322_4 = \_\_\_8$
2	$11010110101111,1011101_2 = \_\_\_8$	$3131232,3202_4 = \_\_\_8$
3	$11010110101111,1011101_2 = \_\_\_16$	$66147,55_8 = \_\_\_4$
4	$123321020,1203_4 = \_\_\_16$	$33556,704_8 = \_\_\_16$
5	$11011101101110,1110001_2 = \_\_\_4$	$12301213,231_4 = \_\_\_8$
6	$11011101101110,1110001_2 = \_\_\_8$	$3333000,132_4 = \_\_\_8$
7	$11011101101110,1110001_2 = \_\_\_16$	$1003320,321_4 = \_\_\_8$
8	$1200033302,11121_4 = \_\_\_16$	$1320013,333_4 = \_\_\_8$
9	$11011000110011 1,101101_2 = \_\_\_4$	$10370,71_8 = \_\_\_16$
10	$11011000110011 1,101101_2 = \_\_\_8$	$17007,77_8 = \_\_\_16$
11	$11011000110011 1,101101_2 = \_\_\_16$	$13652,254_8 = \_\_\_16$
12	$23111112213,213231_4 = \_\_\_16$	$37700,36_8 = \_\_\_4$
13	$111111110000 00,01111_2 = \_\_\_4$	$10370,71_8 = \_\_\_4$
14	$111111110000 00,01111_2 = \_\_\_8$	$17007,77_8 = \_\_\_4$
15	$111111110000 00,01111_2 = \_\_\_16$	$13652,254_8 = \_\_\_4$
16	$11222233311,11223_4 = \_\_\_16$	$15327,564_8 = \_\_\_16$
17	$1000011111000,111001_2 = \_\_\_4$	$1ad7,ba_{16} = \_\_\_8$
18	$1000011111000,111001_2 = \_\_\_8$	$376e,e2_{16} = \_\_\_8$
19	$1000011111000,111001_2 = \_\_\_16$	$6c67,b4_{16} = \_\_\_8$
20	$12312323321,2332_4 = \_\_\_16$	$1132222,1112_4 = \_\_\_8$
21	$1111000000111,111111_2 = \_\_\_4$	$15327,564_8 = \_\_\_4$
22	$1111000000111,111111_2 = \_\_\_8$	$33556,704_8 = \_\_\_4$
23	$1111000000111,111111_2 = \_\_\_16$	$66147,55_8 = \_\_\_16$
24	$10000223311,12332_4 = \_\_\_16$	$37700,36_8 = \_\_\_16$
25	$1011110101010,0101011_2 = \_\_\_4$	$3fc0,78_{16} = \_\_\_8$
26	$1011110101010,0101011_2 = \_\_\_8$	$10f8,e4_{16} = \_\_\_8$
27	$1011110101010,0101011_2 = \_\_\_16$	$1e07,fc_{16} = \_\_\_8$
28	$1111122222,33333_4 = \_\_\_16$	$17aa,56_{16} = \_\_\_8$
29	$10100011111,110101_2 = \_\_\_4$	$3333000,132_4 = \_\_\_8$
30	$111101010101,10101_2 = \_\_\_8$	$13652,254_8 = \_\_\_4$

Таблица 2.2

K	Представления чисел															
2	0	1	10	11	100	101	110	111	1000	1001	1010	1011	1100	1101	1110	1111
3	0	1	2	10	11	12	20	21	22	100	101	102	110	111	112	120
4	0	1	2	3	10	11	12	13	20	21	22	23	30	31	32	33
5	0	1	2	3	4	10	11	12	13	14	20	21	22	23	24	30
6	0	1	2	3	4	5	10	11	12	13	14	15	20	21	22	23
8	0	1	2	3	4	5	6	7	10	11	12	13	14	15	16	17
10	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
16	0	1	2	3	4	5	6	7	8	9	*a*	*b*	*c*	*d*	*e*	*f*

Перевод $10 \to P$ (из 10-ичной СС в P-ичную) осуществляют также средствами десятичной арифметики, причём целую и дробную части числа переводят отдельно. Целую часть переводят делением на P нацело с остатком до достижения частного, равного нулю. Дробную часть переводят умножением на P с отделением целой части произведения до достижения дробной части произведения, равной нулю. Каждый остаток от деления и каждую целую часть произведения представляют одной P-ичной цифрой. Результаты перевода целой и дробной частей объединяют.

Если необходимо выполнить перевод $P \to Q$ при $P \neq 10$ и $Q \neq 10$, то действуют по схеме $P \to 10 \to Q$, т.е. последовательно выполняют переводы $P \to 10$ и $10 \to Q$. Однако, когда значения P и Q определённым образом взаимосвязаны, перевод можно выполнить с меньшими трудозатратами. Так, например, если $P^1 = Q^n$ (n – натуральное число), то для выполнения перевода достаточно каждую P-ичную цифру исходного числа представить соответствующим n-значным Q-ичным числом. Если же $P^n = Q^1$, то при переводе каждое выделенное в исходном представлении n-значное число можно заменить соответствующей Q-ичной цифрой. Наконец, если $P^1 = R^n$ и $R^m = Q^1$ (R, n, m – натуральные числа), то перевод целесообразно осуществить по схеме $P \to R \to Q$ с последовательным применением двух рассмотренных выше способов.

Пример выполнения задания

1а. Выполнить перевод $10111{,}111_2 = \_\_{}_{10}$.

Решение.

Для выполнения перевода $2 \to 10$ (из двоичной СС в десятичную) исходную запись числа представим в виде полинома, заменим в нём все двоичные представления чисел десятичными (см. табл. 2.2) и вычислим значение этого полинома средствами десятичной арифметики:

$$\begin{matrix} 4\ 3\ 2\ 1\ 0 & -1-2-3 \\ 1\ 0\ 1\ 1\ 1\ ,\ 1\ 1\ 1_{\,2} & \end{matrix} = 1_2 \cdot 10_2^4 + 0_2 \cdot 10_2^3 + 1_2 \cdot 10_2^2 + 1_2 \cdot 10_2^1 + 1_2 \cdot 10_2^0 +$$

$$+ 1_2 \cdot 10_2^{-1} + 1_2 \cdot 10_2^{-2} + 1_2 \cdot 10_2^{-3} = 1 \cdot 2^4 + 0 \cdot 2^3 + 1 \cdot 2^2 + 1 \cdot 2^1 + 1 \cdot 2^0 + 1 \cdot 2^{-1} +$$

$$+ 1 \cdot 2^{-2} + 1 \cdot 2^{-3} = 1 \cdot 16 + 0 + 1 \cdot 4 + 1 \cdot 2 + 1 \cdot 1 + 1 \cdot 1/2 + 1 \cdot 1/4 + 1 \cdot 1/8 =$$

$$= 16 + 4 + 2 + 1 + 0{,}5 + 0{,}25 + 0{,}125 = 23{,}875.$$

Ответ: $23{,}875_{10}$.

1б. Выполнить перевод $214{,}24_5 = \_\_{}_{10}$.

Решение.

Для выполнения перевода $5 \to 10$ (из пятеричной СС в десятичную) исходную запись числа представим в виде полинома, заменим в нём все пятеричные представления чисел десятичными (см. табл. 2.2) и вычислим значение полинома средствами десятичной арифметики:

$$\begin{matrix} 2\ 1\ 0 & -1-2 \\ 2\ 1\ 4\ ,\ 2\ 4_{\,5} & \end{matrix} = 2_5 \cdot 10_5^2 + 1_5 \cdot 10_5^1 + 4_5 \cdot 10_5^0 + 2_5 \cdot 10_5^{-1} + 4_5 \cdot 10_5^{-2} =$$

$$= 2 \cdot 5^2 + 1 \cdot 5^1 + 4 \cdot 5^0 + 2 \cdot 5^{-1} + 4 \cdot 5^{-2} = 2 \cdot 25 + 1 \cdot 5 + 4 \cdot 1 + 2 \cdot 1/5 +$$

$$+ 4 \cdot 1/25 = 50 + 5 + 4 + 0{,}4 + 0{,}16 = 59{,}56.$$

Ответ: $59{,}56_{10}$.

1в. Выполнить перевод $1f9{,}c_{16} = \_\_{}_{10}$.

Решение.

Для выполнения перевода $16 \to 10$ (из шестнадцатеричной СС в десятичную) исходную запись числа представим в виде полинома, заменим в нём все шестнадцатеричные представления чисел десятичными (см. табл. 2.2) и вычислим значение полинома средствами десятичной арифметики:

$$2\ 1\ 0\ \ -1$$
$$1\ f\ 9\ ,\ c_{16} = 1_{16}\cdot 10^{2}_{16} + f_{16}\cdot 10^{1}_{16} + 9_{16}\cdot 10^{0}_{16} + c_{16}\cdot 10^{-1}_{16} =$$

$$= 1\cdot 16^2 + 15\cdot 16^1 + 9\cdot 16^0 + 12\cdot 16^{-1} = 1\cdot 256 + 15\cdot 16 + 9\cdot 1 + 12\cdot 1/16 =$$
$$= 256 + 240 + 9 + 0{,}75 = 505{,}75.$$

Ответ: $505{,}75_{10}$.

2а. Выполнить перевод $75{,}515625_{10} = \_\_{}_4$.

Решение.

Перевод $10 \to 4$ (из десятичной СС в четверичную) осуществим средствами десятичной арифметики, причём целую и дробную часть числа переведём отдельно.

Целую часть переведём делением на 4 нацело с остатком до достижения частного, равного нулю, и каждый остаток от деления представим четверичной цифрой:

7 5 | 4	1 8 | 4	4 | 4	1 | 4
4 1 8	1 6 4	4 1	0 0
3 5	2 = 2_4	0 = 0_4	1 = 1_4
3 2			
3 = 3_4			

$$75_{10} = 1023_4.$$

Дробную часть переведём умножением на 4 с отделением целой части произведения до достижения дробной части произведения, равной нулю, и каждую целую часть полученных произведений представим четверичной цифрой:

$0{,}515625 \cdot 4 = 2{,}0625,$	$2 = 2_4,$
$0{,}0625 \cdot 4 = 0{,}25,$	$0 = 0_4,$
$0{,}25 \cdot 4 = 1{,}0,$	$1 = 1_4,$

$$0{,}515625_{10} = 0{,}201_4.$$

Объединив результаты перевода целой и дробной части, получим $75{,}515625_{10} = 1023{,}201_4$.

Ответ: $1023{,}201_4$.

2б. Выполнить перевод $134{,}453125_{10} = \_\_{}_8$.

Решение.

Перевод $10 \to 8$ (из десятичной СС в восьмеричную) осуществим средствами десятичной арифметики, причём целую и дробную части числа переведём отдельно.

Целую часть переведём делением на 8 нацело с остатком до достижения частного, равного нулю, и каждый остаток от деления представим восьмеричной цифрой:

$$\begin{array}{rrr|l} 1 & 3 & 4 & 8 \\ \hline 8 & & & 1\ 6 \\ \hline 5 & 4 & & \\ 4 & 8 & & \\ \hline 6 & & & \end{array} = 6_8 \qquad \begin{array}{rr|l} 1 & 6 & 8 \\ \hline 1 & 6 & 2 \\ \hline 0 & & \end{array} = 0_8 \qquad \begin{array}{rr|l} 2 & & 8 \\ \hline 0 & 0 & \\ \hline 2 & & \end{array} = 2_8$$

$$134_{10} = 206_8.$$

Дробную часть переведём умножением на 8 с отделением целой части произведения до достижения дробной части произведения, равной нулю, и каждую целую часть полученных произведений представим восьмеричной цифрой:

$$0{,}453125 \cdot 8 = 3{,}625, \qquad 3 = 3_8,$$
$$0{,}625 \cdot 8 = 5{,}0, \qquad 5 = 5_8,$$

$$0{,}453125_{10} = 0{,}35_8.$$

Объединив результаты перевода целой и дробной части, получим

$$134{,}453125_{10} = 206{,}35_8.$$

Ответ: $206{,}35_8$.

2в. Выполнить перевод $2983{,}875_{10} = \_\_{}_{16}$.

Решение.

Перевод $10 \rightarrow 16$ (из десятичной СС в восьмеричную) осуществим средствами десятичной арифметики, причём целую и дробную части числа переведём отдельно.

Целую часть переведём делением на 16 нацело с остатком до достижения частного, равного нулю, и каждый остаток от деления представим шестнадцатеричной цифрой:

$$\begin{array}{rrrr|l} 2 & 9 & 8 & 3 & 1\ 6 \\ \hline 1 & 6 & & & 1\ 8\ 6 \\ \hline 1 & 3 & 8 & & \\ 1 & 2 & 8 & & \\ \hline & 1 & 0 & 3 & \\ & & 9 & 6 & \\ \hline & & & 7 & \end{array} = 7_{16} \qquad \begin{array}{rrr|l} 1 & 8 & 6 & 1\ 6 \\ \hline 1 & 6 & & 1\ 1 \\ \hline & 2 & 6 & \\ & 1 & 6 & \\ \hline & 1 & 0 & \end{array} = a_{16} \qquad \begin{array}{rr|l} 1 & 1 & 1\ 6 \\ \hline & 0 & 0 \\ \hline 1 & 1 & \end{array} = b_{16}$$

$$2983_{10} = ba7_{16}.$$

Дробную часть переведём умножением на 16 с отделением целой части произведения до достижения дробной части произведения, равной нулю, и каждую целую часть полученных произведений представим шестнадцатеричной цифрой:

$0{,}875 \cdot 16 = 14{,}0,$ $\qquad 14 = e_{16},$ $\qquad 0{,}875_{10} = 0{,}e_{16}.$

Объединив результаты перевода целой и дробной частей, получим $2983{,}875_{10} = ba7{,}e_{16}$.

Ответ: $ba7{,}e_{16}$.

3а. Выполнить перевод $25701{,}14_8 = \_\_{}_2$.

Решение.

Заметим, что $8^1 = 2^3$. Поэтому при выполнении перевода каждую восьмеричную цифру представим соответствующим трёхзначным двоичным числом (см. табл. 2.1):

$2_8 = 10_2 = 010_2,$ $\quad 5_8 = 101_2,$ $\quad 7_8 = 111_2,$
$0_8 = 0_2 = 000_2,$ $\quad 1_8 = 1_2 = 001_2,$ $\quad 4_8 = 100_2,$

$25701{,}14_8 =$ | 010 | 101 | 111 | 000 | 001 | , | 001 | 100 |$_2$.

Отбросим у полученного двоичного числа слева и справа незначащие нули:

$010101111000001{,}001100_2 = 10101111000001{,}0011_2$.

Ответ: $10101111000001{,}0011_2$.

3б. Выполнить перевод $1b49e{,}0f6_{16} = \_\_{}_4$.

Решение.

Заметим, что $16^1 = 4^2$. Поэтому при выполнении перевода каждую шестнадцатеричную цифру представим соответствующим двузначным четверичным числом (см. табл. 2.1):

$1_{16} = 1_4 = 01_4,$ $\quad b_{16} = 23_4,$ $\quad 4_{16} = 10_4,$ $\quad 9_{16} = 21_4,$
$e_{16} = 32_4,$ $\quad 0_{16} = 0_4 = 00_4,$ $\quad f_{16} = 33_4,$ $\quad 6_{16} = 12_4,$

$1b49e{,}0f6_{16} =$ | 01 | 23 | 10 | 21 | 32 | , | 00 | 33 | 12 |$_4$.

Отбросим у полученного двоичного числа слева незначащий нуль:

$0123102132{,}003312_4 = 123102132{,}003312_4$.

Ответ: $123102132{,}003312_4$.

4. Выполнить перевод $11000111001101{,}1111001_2 = \_\_{}_{16}$.

Решение.

Заметим, что $2^4 = 16^1$. Поэтому добавим слева и справа к заданному двоичному числу незначащие нули до достижения в целой и дробной частях количеств цифр, кратных четырём:

$11000111001101{,}1111001_2 = 0011000111001101{,}11110010_2$.

Выделим в полученном представлении четырёхзначные двоичные числа:

$$0011000111001101,11110010_2 = \boxed{0011}\boxed{0001}\boxed{1100}\boxed{1101}, \boxed{1111}\boxed{0010}_2.$$

Каждое выделенное четырёхзначное двоичное число заменим одной соответствующей шестнадцатеричной цифрой (см. табл. 2.1):

$0011_2 = 11_2 = 3_{16}$, $0001_2 = 1_2 = 1_{16}$, $1100_2 = c_{16}$,
$1101_2 = d_{16}$, $1111_2 = f_{16}$, $0010_2 = 10_2 = 2_{16}$,

$$\boxed{0011}\boxed{0001}\boxed{1100}\boxed{1101}, \boxed{1111}\boxed{0010}_2 = 31cd,f2_{16}.$$

Ответ: $31cd,f2_{16}$.

5. Выполнить перевод $5a0b,c1_{16} = \_\_{}_8$.

Решение.

Заметим, что $16^1 = 2^4$ и $2^3 = 8^1$. Поэтому перевод осуществим по схеме $16 \to 2 \to 8$.

Переведём исходное число в двоичную систему счисления путём замены каждой шестнадцатеричной цифры соответствующим четырёхзначным двоичным числом (см. табл. 2.1):

$$5a0b,c1_{16} = \boxed{0101}\boxed{1010}\boxed{0000}\boxed{1011}, \boxed{1100}\boxed{0001}_2 =$$

$$= 1011010000001011,11000001_2.$$

Переведём полученное двоичное число в восьмеричную систему счисления путём замены каждого выделенного трёхзначного двоичного числа соответствующей восьмеричной цифрой (см. табл. 2.1):

$$1011010000001011,11000001_2 = 1011010000001011,110000010_2 =$$

$$= \boxed{101}\boxed{101}\boxed{000}\boxed{001}\boxed{011}, \boxed{110}\boxed{000}\boxed{010}_2 = 55013,602_8.$$

Ответ: $55013,602_8$.

Лабораторная работа № 3

АРИФМЕТИЧЕСКИЕ ОПЕРАЦИИ В ПОЗИЦИОННЫХ СИСТЕМАХ СЧИСЛЕНИЯ

Цель: приобретение навыков выполнения арифметических операций над числами в позиционных системах счисления с различными основаниями.

З*адание*. Выполнить операции сложения, вычитания, умножения и деления над числами в заданной системе счисления (табл. 3.1).

Таблица 3.1

Вариант	Задание	
	1	2
1	$1022{,}12_3 + 212{,}2_3 = \_\_{}_3$	$2210{,}2_3 - 121{,}12_3 = \_\_{}_3$
2	$1232{,}12_4 + 212{,}2_4 = \_\_{}_4$	$2020{,}2_4 - 321{,}03_4 = \_\_{}_4$
3	$1234{,}32_5 + 141{,}02_5 = \_\_{}_5$	$4041{,}04_5 - 312{,}13_5 = \_\_{}_5$
4	$2345{,}23_6 + 242{,}32_6 = \_\_{}_6$	$5221{,}13_6 - 232{,}3_6 = \_\_{}_6$
5	$2102{,}21_3 + 122{,}1_3 = \_\_{}_3$	$10002{,}01_3 - 122{,}1_3 = \_\_{}_3$
6	$1322{,}31_4 + 112{,}3_4 = \_\_{}_4$	$2101{,}21_4 - 112{,}3_4 = \_\_{}_4$
7	$2143{,}23_5 + 232{,}12_5 = \_\_{}_5$	$2430{,}4_5 - 232{,}12_5 = \_\_{}_5$
8	$3253{,}33_6 + 123{,}13_6 = \_\_{}_6$	$3420{,}5_6 - 123{,}13_6 = \_\_{}_6$
9	$1121{,}22_3 + 112{,}2_3 = \_\_{}_3$	$2011{,}12_3 - 112{,}2_3 = \_\_{}_3$
10	$2231{,}13_4 + 223{,}2_4 = \_\_{}_4$	$3120{,}33_4 - 223{,}2_4 = \_\_{}_4$
11	$2343{,}12_5 + 231{,}41_5 = \_\_{}_5$	$3130{,}03_5 - 231{,}41_5 = \_\_{}_5$
12	$1552{,}12_6 + 123{,}5_6 = \_\_{}_6$	$2120{,}02_6 - 123{,}5_6 = \_\_{}_6$
13	$2201{,}21_3 + 121{,}1_3 = \_\_{}_3$	$2101_3 - 201{,}12_3 = \_\_{}_3$
14	$2133{,}32_4 + 132{,}21_4 = \_\_{}_4$	$3023{,}02_4 - 213{,}2_4 = \_\_{}_4$
15	$2233{,}44_5 + 321{,}12_5 = \_\_{}_5$	$2130{,}4_5 - 233{,}32_5 = \_\_{}_5$
16	$2453{,}32_6 + 103{,}23_6 = \_\_{}_6$	$2015{,}4_6 - 122{,}12_6 = \_\_{}_6$
17	$2121{,}2_3 + 122{,}22_3 = \_\_{}_3$	$2111{,}1_3 - 121{,}21_3 = \_\_{}_3$
18	$1123{,}21_4 + 311{,}2_4 = \_\_{}_4$	$10031{,}2_4 - 112{,}31_4 = \_\_{}_4$
19	$3324{,}13_5 + 130{,}41_5 = \_\_{}_5$	$3102{,}11_5 - 212{,}2_5 = \_\_{}_5$
20	$3253{,}41_6 + 234{,}5_6 = \_\_{}_6$	$3043{,}04_6 - 203{,}45_6 = \_\_{}_6$
21	$2012{,}01_3 + 121{,}12_3 = \_\_{}_3$	$2012{,}02_3 - 212{,}2_3 = \_\_{}_3$
22	$1033{,}11_4 + 321{,}03_4 = \_\_{}_4$	$2110{,}32_4 - 212{,}2_4 = \_\_{}_4$
23	$3223{,}41_5 + 312{,}13_5 = \_\_{}_5$	$1430{,}34_5 - 141{,}02_5 = \_\_{}_5$
24	$4544{,}43_6 + 232{,}3_6 = \_\_{}_6$	$3031{,}55_6 - 242{,}32_6 = \_\_{}_6$
25	$1122{,}11_3 + 201{,}12_3 = \_\_{}_3$	$10100{,}01_3 - 121{,}1_3 = \_\_{}_3$
26	$2203{,}22_4 + 213{,}2_4 = \_\_{}_4$	$2332{,}13_4 - 132{,}21_4 = \_\_{}_4$
27	$1342{,}03_5 + 233{,}32_5 = \_\_{}_5$	$3110{,}11_5 - 321{,}12_5 = \_\_{}_5$
28	$1453{,}24_6 + 122{,}12_6 = \_\_{}_6$	$3000{,}55_6 - 103{,}23_6 = \_\_{}_6$
29	$1212{,}12_3 + 121{,}21_3 = \_\_{}_3$	$10021{,}12_3 - 122{,}22_3 = \_\_{}_3$
30	$3312{,}23_4 + 112{,}31_4 = \_\_{}_4$	$2101{,}01_4 - 311{,}2_4 = \_\_{}_4$

Продолжение табл. 3.1

Вариант	Задание	
	3	4
1	$1212{,}1_3 \times 0{,}22_3 = \_\_{}_3$	$11122{,}111_3 : 2{,}1_3 = \_\_{}_3$
2	$1232{,}12_4 \times 2{,}2_4 = \_\_{}_4$	$2022{,}303_4 : 1{,}3_4 = \_\_{}_4$
3	$1233{,}4_5 \times 2{,}1_5 = \_\_{}_5$	$10233{,}14_5 : 3{,}3_5 = \_\_{}_5$
4	$1235{,}5_6 \times 1{,}2_6 = \_\_{}_6$	$41152{,}53_6 : 5{,}1_6 = \_\_{}_6$
5	$1122{,}12_3 \times 2{,}2_3 = \_\_{}_3$	$100022{,}101_3 : 10{,}2_3 = \_\_{}_3$
6	$1322{,}31_4 \times 1{,}2_4 = \_\_{}_4$	$11300{,}022_4 : 2{,}1_4 = \_\_{}_4$
7	$3124{,}1_5 \times 1{,}2_5 = \_\_{}_5$	$24231{,}02_5 : 4{,}1_5 = \_\_{}_5$
8	$3451{,}2_6 \times 2{,}1_6 = \_\_{}_6$	$11145{,}52_6 : 1{,}5_6 = \_\_{}_6$
9	$2212{,}22_3 \times 10{,}2_3 = \_\_{}_3$	$11101{,}211_3 : 2{,}2_3 = \_\_{}_3$
10	$2231{,}13_4 \times 2{,}1_4 = \_\_{}_4$	$2320{,}032_4 : 1{,}2_4 = \_\_{}_4$
11	$2312{,}3_5 \times 3{,}1_5 = \_\_{}_5$	$4304{,}42_5 : 1{,}2_5 = \_\_{}_5$
12	$2235{,}3_6 \times 1{,}3_6 = \_\_{}_6$	$12131{,}52_6 : 2{,}1_6 = \_\_{}_6$
13	$1102{,}21_3 \times 0{,}21_3 = \_\_{}_3$	$101120{,}121_3 : 10{,}2_3 = \_\_{}_3$
14	$2133{,}32_4 \times 3{,}3_4 = \_\_{}_4$	$12012{,}033_4 : 2{,}1_4 = \_\_{}_4$
15	$1324{,}2_5 \times 1{,}3_5 = \_\_{}_5$	$13224{,}13_5 : 3{,}1_5 = \_\_{}_5$
16	$4521{,}4_6 \times 3{,}1_6 = \_\_{}_6$	$3355{,}13_6 : 1{,}3_6 = \_\_{}_6$
17	$2012{,}22_3 \times 2{,}1_3 = \_\_{}_3$	$1010{,}0111_3 : 0{,}21_3 = \_\_{}_3$
18	$1123{,}21_4 \times 3{,}1_4 = \_\_{}_4$	$21113{,}202_4 : 3{,}3_4 = \_\_{}_4$
19	$3221{,}2_5 \times 2{,}2_5 = \_\_{}_5$	$2333{,}01_5 : 1{,}3_5 = \_\_{}_5$
20	$3351{,}4_6 \times 1{,}4_6 = \_\_{}_6$	$23301{,}14_6 : 3{,}1_6 = \_\_{}_6$
21	$1222{,}21_3 \times 2{,}1_3 = \_\_{}_3$	$110010{,}221_3 : 20{,}1_3 = \_\_{}_3$
22	$1033{,}11_4 \times 1{,}3_4 = \_\_{}_4$	$21220{,}232_4 : 2{,}2_4 = \_\_{}_4$
23	$1232{,}3_5 \times 3{,}3_5 = \_\_{}_5$	$2334{,}23_5 : 1{,}4_5 = \_\_{}_5$
24	$4522{,}3_6 \times 5{,}1_6 = \_\_{}_6$	$10420{,}3_6 : 2{,}2_6 = \_\_{}_6$
25	$2112{,}12_3 \times 10{,}2_3 = \_\_{}_3$	$1122{,}202_3 : 0{,}22_3 = \_\_{}_3$
26	$2203{,}22_4 \times 2{,}1_4 = \_\_{}_4$	$10103{,}33_4 : 2{,}2_4 = \_\_{}_4$
27	$3212{,}2_5 \times 4{,}1_5 = \_\_{}_5$	$3201{,}14_5 : 2{,}1_5 = \_\_{}_5$
28	$3552{,}4_6 \times 1{,}5_6 = \_\_{}_6$	$1531{,}44_6 : 1{,}2_6 = \_\_{}_6$
29	$1220{,}21_3 \times 20{,}1_3 = \_\_{}_3$	$12011{,}202_3 : 2{,}1_3 = \_\_{}_3$
30	$3312{,}23_4 \times 2{,}2_4 = \_\_{}_4$	$10221{,}211_4 : 3{,}1_4 = \_\_{}_4$

Основные положения

Арифметические операции над числами в любой позиционной СС выполняются по общим правилам с использованием соответствующих таблиц сложения и умножения.

Таблица сложения в K-ичной СС имеет размер $(K+1)\times(K+1)$, включая заголовки строк и столбцов. Каждой строке таблицы сложения взаимно однозначно сопоставляется первое слагаемое из диапазона $[0, K-1]$, а каждому столбцу – второе слагаемое из того же диапазона. Заполняется таблица сложения следующим образом: в первую строку таблицы необходимо записать числа 0, 1, 2, ..., $(K-1)$; во вторую – числа 1, 2, 3, ..., K; в третью – числа 2, 3, 4, ..., $(K+1)$ и т.д. до тех пор, пока не будет заполнена вся таблица.

Таблица умножения строится аналогично. В таблице умножения целесообразно заполнить сначала её нижнюю треугольную часть, включая главную диагональ: в первый столбец таблицы записать нули, а в каждую ячейку (i, j) – результат сложения значения в ячейке $(i, j-1)$ со значением сомножителя, соответствующего i-й строке таблицы. Затем выполнить симметричное отображение матрицы относительно главной диагонали, поскольку операция умножения удовлетворяет закону коммутативности.

Пример выполнения задания

1. Выполнить операцию сложения над числами в заданной системе счисления $1202{,}12_3 + 112{,}02_3 = \_\_{}_3$.

Решение.

Построим таблицу сложения в троичной системе счисления:

$+_3$	0	1	2
0	0	1	2
1	1	2	10
2	2	10	11

С её помощью выполним сложение:

$$\begin{array}{cccccccc} & {\scriptstyle 1} & & {\scriptstyle 1} & & & {\scriptstyle 1} & \\ & 1 & 2 & 0 & 2 & , & 1 & 2 \\ + & & & & & & & \\ & & 1 & 1 & 2 & , & 0 & 2 \\ \hline 2 & 0 & 2 & 1 & , & 2 & 1 & \end{array}$$

$2+2=11=10+1$

$1+1+0=2+0=2$

$2+2=11=10+1$

$1+0+1=1+1=2$

$2+1=10=10+0$

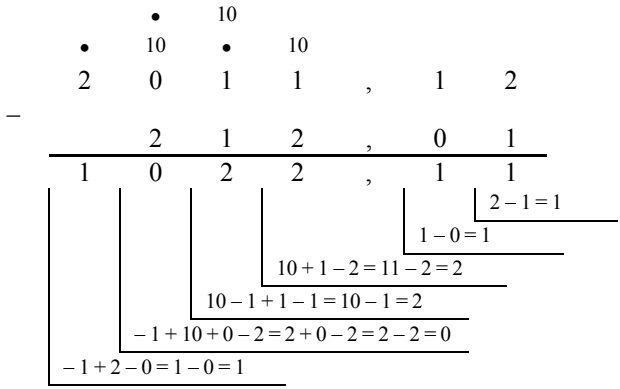
$1+1=2$

Ответ: $2021{,}21_3$.

2. Выполнить операцию вычитания над числами в заданной системе счисления $2011,12_3 - 212,01_3 = \_\_{}_3$.

Решение.

Вычитание выполним с помощью таблицы сложения в троичной системе счисления:



Ответ: $1022,11_3$.

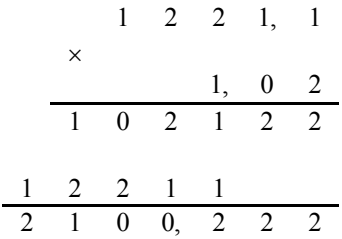
3. Выполнить операцию умножения над числами в заданной системе счисления $1221,1_3 \times 1,02_3 = \_\_{}_3$.

Решение.

С помощью таблицы сложения в троичной системе счисления построим троичную таблицу умножения:

$\times_3$	0	1	2
0	0	0	0
1	0	1	2
2	0	2	11

С её помощью выполним умножение:



$2 \times 1 = 2$

$2 \times 1 = 2$

$2 \times 2 = 11 = 10 + 1$

$1 + 2 \times 2 = 1 + 11 = 12 = 10 + 2$

$1 + 2 \times 1 = 1 + 2 = 10 = 10 + 0$

$1 + 0 = 1$

Ответ: $2100,222_3$.

4. Выполнить операцию деления над числами в заданной системе счисления $2000,11_3 : 1,2_3 = \_\_{}_3$.

Решение.

Сначала умножим делимое и делитель на 10_3 для того, чтобы делить на целое число:

$2000{,}11_3 : 1{,}2_3 = (2000{,}11_3 \times 10_3) : (1{,}2_3 \times 10_3) = 20001{,}1_3 : 12_3$.

Затем выполним подготовительные операции умножения для того, чтобы легче было подбирать цифры получаемого частного:

$12 \times 1 = 12$, $12 \times 2 = 101$.

И, наконец, выполним деление:

```
  2  0  0  0  1,  1 | 1  2
–                    1  0  1  2,  2
  1  2
  ---
     1  0  0
  –
        1  2
        ---
        1  1  1
     –
        1  0  1
        ---
           1  0  1
        –
           1  0  1
           ------
              0
```

Ответ: $1012{,}2_3$.

Лабораторная работа № 4

ПРЕДСТАВЛЕНИЕ ЧИСЕЛ В ПАМЯТИ ЭВМ

Цель: изучение беззнакового и знакового представлений целых чисел, а также представления чисел в формате с плавающей точкой в памяти ЭВМ.

Задание.

1. Получить однобайтное беззнаковое представление десятичного числа.
2. Получить десятичное число по его однобайтному беззнаковому представлению.
3. Получить однобайтное знаковое представление десятичного числа.
4. Получить десятичное число по его однобайтному знаковому представлению.
5. Получить четырёхбайтное представление десятичного числа в формате с плавающей точкой.
6. Получить десятичное число по его четырёхбайтному представлению в формате с плавающей точкой (табл. 4.1).

Таблица 4.1

Вариант	Задание			
	1	2	3	4
1	189	00001100	–53	11111001
2	66	00010001	–23	11001100
3	111	00001001	–14	10010011
4	6	01000101	4	11100010
5	243	01001101	10	10000110
6	7	00000010	69	11101010
7	219	00100001	–91	00010100
8	123	10010110	–42	00001001
9	209	01110011	104	11111100
10	170	11100111	–70	11111000
11	242	01011110	40	10101110
12	162	10001100	85	00110110
13	252	01000011	–18	00111011
14	58	01010000	82	01010111
15	69	01100011	–90	01011111
16	220	10110010	55	01010011
17	222	01110100	14	00101000
18	13	11101000	–113	10011101
19	246	00110001	–55	10010100
20	203	00010101	13	01100100
21	105	00111110	–55	11011100
22	21	11101000	–82	11101100
23	192	01001010	8	01110011
24	22	00011001	–22	01110000
25	227	00100011	–45	11000100
26	44	11100101	96	01110100
27	24	10000101	–91	00010001
28	162	01100000	101	01111100
29	128	00010000	69	00100111
30	125	11110010	108	00001001

Продолжение табл. 4.1

Вариант	Задание	
	5	6
1	3,5	11000100011100000000000000000000
2	–0,001953125	01000100110100000000000000000000
3	–5	01000010111000000000000000000000
4	–256	00111111010000000000000000000000
5	32	00111111111000000000000000000000
6	3072	10111100000000000000000000000000
7	20	01000011011000000000000000000000
8	0,015625	10111111101000000000000000000000
9	0,046875	01000101001100000000000000000000
10	–112	10111101100000000000000000000000
11	–16	01000011001000000000000000000000
12	0,25	11000010000000000000000000000000
13	–3,5	00111111100000000000000000000000
14	–60	01000011100000000000000000000000
15	–20	00111101010000000000000000000000
16	–16	00111100100000000000000000000000
17	6144	10111111100000000000000000000000
18	–128	11000110010000000000000000000000
19	112	11000010000100000000000000000000
20	–43,25	00111111101100000000000000000000
21	22	00111100101000000000000000000000
22	–0,0625	11000001000000000000000000000000
23	–6,5	11000001111100000000000000000000
24	–0,9375	11000001110100000000000000000000
25	1024	01000000001100000000000000000000
26	3,75	00111100110000000000000000000000
27	0,5	10111100010000000000000000000000
28	0,171875	10111110100000000000000000000000
29	–384	11000011110100000000000000000000
30	5	11000101111100000000000000000000

Основные положения

Числа, как и любая другая информация, представляются в памяти ЭВМ в виде цифровых двоичных кодов.

Однобайтное беззнаковое представление обеспечивает хранение 256 десятичных целых чисел из диапазона от 0 до 255 включительно. Для получения такого представления десятичного числа из приведённого выше диапазона необходимо перевести это число в 2-ичную СС и при необходимости полученный двоичный код дополнить слева нулями до 8 цифр. Для определения десятичного числа по его беззнаковому представлению необходимо перевести заданное 2-ичное число в десятичную СС.

Знаковое представление в одном байте памяти тоже обеспечивает хранение 256 десятичных целых чисел, но уже из диапазона от –128 до 127 включительно. При этом неотрицательные числа (положительные числа и нуль) кодируются таким же образом, как и при беззнаковом представлении, а для получения знакового представления отрицательного числа необходимо: определить абсолютную величину заданного числа; уменьшить полученное число на один; определить беззнаковое представление последнего числа и, наконец, инвертировать полученный двоичный код (каждый 0 заменить на 1 и каждую 1 заменить на 0). Для получения десятичного числа по его знаковому представлению необходимо: если двоичный код начинается с цифры 0, то действовать также, как и при беззнаковом представлении; в противном случае – инвертировать двоичный код, перевести полученное двоичное число в десятичную СС, увеличить полученное десятичное представление на один и, наконец, присоединить к последнему числу знак «минус».

При четырёхбайтном представлении числа в формате с плавающей точкой первый из 32 бит хранит знак числа (плюс кодируется нулём, а минус – единицей); биты со второго по девятый – порядок числа, увеличенный на 128; оставшиеся 23 бита – нормализованную мантиссу числа без первой после запятой цифры, поскольку в такой мантиссе эта цифра всегда 1. Для получения четырёхбайтного представления десятичного числа в формате с плавающей точкой необходимо: перевести заданное число (без учёта знака минус для отрицательных чисел) в двоичную СС; определить экспоненциальную форму двоичного числа с нормализованной мантиссой и соответствующим порядком; полученный порядок увеличить на 128_{10} (10000000_2) и заполнить биты искомого представления. Для получения десятичного числа по его четырёхбайтному представлению в формате с плавающей точкой необходимо: определить знак числа; выписать порядок и уменьшить его на 128_{10} (10000000_2); определить нормализованную мантиссу; записать экспоненциальную форму числа и получить его десятичное представление.

Пример выполнения задания

1. Получить однобайтное беззнаковое представление десятичного числа 98.

Решение.

Для получения однобайтного беззнакового представления заданного числа переведём его в 2-ичную СС и при необходимости полученный двоичный код дополним слева нулями до 8 цифр:

$98_{10} = 1100010_2 = 01100010_2$.

Ответ: 01100010.

2. Получить десятичное число по его однобайтному беззнаковому представлению 00100101.

Решение.

Определим десятичное число по его беззнаковому представлению необходимо путём перевода заданного 2-ичного числа в десятичную СС:

$00100101_2 = 37_{10}$.

Ответ: 37.

3а. Получить однобайтное знаковое представление десятичного числа 51.

Решение.

Поскольку заданное число является неотрицательным, кодируем его таким же образом, как и при беззнаковом представлении:

$51_{10} = 110011_2 = 00110011_2$.

Ответ: 00110011.

3б. Получить однобайтное знаковое представление десятичного числа −34.

Решение.

Поскольку заданное число является отрицательным, определим его абсолютное значение, уменьшим это значение на один, получим беззнаковое представление последнего числа и, наконец, выполним инверсию двоичного кода:

$-34_{10} \to 34_{10} \to 33_{10} \to 00100001_2 \to 11011110_2$.

Ответ: 11011110.

4а. Получить десятичное число по его однобайтному знаковому представлению 01001101.

Решение.

Поскольку двоичный код начинается с цифры 0, действуем также, как и при беззнаковом представлении:

$01001101_2 = 77_{10}$.

Ответ: 77.

4б. Получить десятичное число по его однобайтному знаковому представлению 10110101.

Решение.

Поскольку двоичный код начинается с цифры 1, инвертируем его, переведём полученное двоичное число в десятичную СС, увеличим десятичное представление на один и, наконец, присоединим к последнему числу знак «минус»:

$10110101_2 \rightarrow 01001010_2 \rightarrow 74_{10} \rightarrow 75_{10} \rightarrow -75_{10}$.

Ответ: –75.

5а. Получить четырёхбайтное представление в формате с плавающей точкой десятичного числа $-18{,}125_{10}$.

Решение.

Поскольку заданное число является отрицательным, первый бит представления будет равен 1.

Переведём заданное число без учёта знака «минус» в двоичную СС:

$18{,}125_{10} = 10010{,}001_2$.

Представим полученное двоичное число в экспоненциальной форме с нормализованной мантиссой и соответствующим порядком:

$10010{,}001_2 = 0{,}10010001_2 \cdot 10_2^{5_{10}} = 0{,}10010001_2 \cdot 10_2^{101_2}$.

Полученный порядок увеличим на 128_{10} (10000000_2):

$101_2 + 10000000_2 = 10000101_2$.

Заполним биты искомого представления, записав в первый бит 1, в биты со второго по девятый – 10000101, в биты с десятого по тридцать второй – 0010001 (цифры дробной части нормализованной мантиссы без первой единицы) и далее нули до конца:

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32
1	1	0	0	0	0	1	0	1	0	0	1	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Ответ: 11000010100100010000000000000000.

5б. Получить четырёхбайтное представление в формате с плавающей точкой десятичного числа $0{,}078125_{10}$.

Решение.

Поскольку заданное число является положительным, первый бит представления будет равен 0.

Переведём заданное число в двоичную СС:

$0{,}078125_{10} = 0{,}000101_2$.

Представим полученное двоичное число в экспоненциальной форме с нормализованной мантиссой и соответствующим порядком:

$0{,}000101_2 = 0{,}101_2 \cdot 10_2^{-3_{10}} = 0{,}101_2 \cdot 10_2^{-11_2}$.

Полученный порядок увеличим на 128_{10} (10000000_2):

$-11_2 + 10000000_2 = 10000000_2 - 11_2 = 1111101_2 = 01111101_2$.

Заполним биты искомого представления, записав в первый бит 0, в биты со второго по девятый – 01111101, в биты с десятого по тридцать второй – 01 (цифры дробной части нормализованной мантиссы без первой единицы) и далее нули до конца:

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32
0	0	1	1	1	1	1	0	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Ответ: 00111110101000000000000000000000.

6а. Получить десятичное число по его четырёхбайтному представлению в формате с плавающей точкой 01000100000100000000000000000000.

Решение.

Поместим заданное представление в таблицу

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32
0	1	0	0	0	1	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

В первом бите находится 0, следовательно, представленное число является положительным.

Получим нормализованную мантиссу экспоненциальной формы представленного числа, записав 0,1 и дополнив эту запись цифрами, представленными в битах с 10-го по 32-й:

$0{,}10010000000000000000000_2 = 0{,}1001_2$.

Определим порядок экспоненциальной формы представленного числа, выписав биты со 2-го по 9-й и уменьшив полученное двоичное число на $128_{10} = 10000000_2$:

$10001000_2 - 10000000_2 = 1000_2$.

Получим искомое десятичное число, записав экспоненциальную форму с нормализованной мантиссой с учётом знака «плюс»:

$0{,}1001_2 \cdot 10_2^{1000_2} = 0{,}1001_2 \cdot 10_2^{8_{10}} = 10010000_2 = 144_{10}$.

Ответ: 144.

6б. Получить десятичное число по его четырёхбайтному представлению в формате с плавающей точкой 10111101011000000000000000000000.

Решение.

Поместим заданное представление в таблицу

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32
1	0	1	1	1	1	0	1	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

В первом бите находится 1, следовательно, представленное число является отрицательным.

Получим нормализованную мантиссу экспоненциальной формы представленного числа, записав 0,1 и дополнив эту запись цифрами, представленными в битах с 10-го по 32-й:

$0{,}11100000000000000000000_2 = 0{,}111_2$.

Определим порядок экспоненциальной формы представленного числа, выписав биты со 2-го по 9-й и уменьшив полученное двоичное число на $128_{10} = 10000000_2$:

$01111010_2 - 10000000_2 = -110_2$.

Получим искомое десятичное число, записав экспоненциальную форму с нормализованной мантиссой с учётом знака «минус»:

$$-0{,}111_2 \cdot 10_2^{-110_2} = -0{,}111_2 \cdot 10_2^{-6_{10}} = -0{,}000000111_2 =$$

$$= -0{,}013671875_{10}.$$

Ответ: – 0,013671875.

Лабораторная работа № 5

СИНТЕЗ ВЫЧИСЛИТЕЛЬНЫХ СХЕМ

Цель: приобретение умений составления и минимизации логических выражений в классе дизъюнктивных нормальных форм (ДНФ), а также построения структурных схем для вычисления их значений.

З*адание*. На плоскости *x*0*y* заданы круг, прямоугольник и треугольник (рис. 5.1). Плоскость разбита линиями, ограничивающими эти фигуры на непересекающиеся множества точек *A*, *B*, *C*, *D*, *E*, *F*, *G* и *H*. Составить совершенную дизъюнктивную нормальную форму (СДНФ) логического выражения, истинного только для точек объединения заданных в табл. 5.1 множеств, минимизировать её и построить соответствующую структурную вычислительную схему.

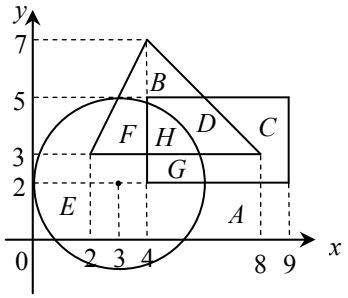


Рис. 5.1

Таблица 5.1

Вариант	Множества точек	Вариант	Множества точек
1	*A*, *B*, *C*	2	*B*, *C*, *D*, *F*, *H*
3	*A*, *E*, *F*	4	*F*, *G*, *H*
5	*D*, *E*, *G*, *H*	6	*A*, *C*, *D*, *E*, *F*, *G*
7	*B*, *D*, *E*, *F*	8	*A*, *B*, *C*, *D*, *E*, *G*
9	*A*, *B*, *C*, *D*, *F*, *G*, *H*	10	*A*, *B*, *C*, *D*, *E*
11	*A*, *F*, *H*	12	*A*, *B*, *C*, *E*, *G*
13	*B*, *C*, *D*, *E*, *F*, *G*, *H*	14	*D*, *E*, *G*
15	*B*, *C*, *F*, *H*	16	*A*, *B*, *D*, *E*, *F*, *H*
17	*A*, *D*, *E*, *F*, *G*	18	*A*, *B*, *C*, *D*, *E*, *F*, *H*
19	*A*, *C*, *D*, *E*, *F*, *G*, *H*	20	*B*, *C*, *D*, *E*, *G*, *H*
21	*A*, *B*, *C*, *E*, *F*, *G*, *H*	22	*A*, *B*, *C*, *D*, *F*, *H*
23	*A*, *B*, *D*, *F*, *H*	24	*A*, *B*, *C*, *F*, *G*
25	*B*, *G*, *H*	26	*B*, *C*, *E*, *F*, *G*, *H*
27	*A*, *B*, *D*, *E*, *F*, *G*, *H*	28	*B*, *C*, *E*, *H*
29	*B*, *D*, *E*, *G*	30	*A*, *B*, *D*, *F*, *G*, *H*

Основные положения

Синтез вычислительных схем для заданных логических выражений включает в себя: составление СДНФ логического выражения, минимизацию логического выражения в классе ДНФ и построение соответствующей структурной схемы для вычисления его значений [1].

Логическими выражениями называются выражения, содержащие переменные, константы, операции отношения и логические операции. *Операциями отношения* являются операции меньше (<), больше (>), меньше или равно (≤), больше или равно (≥), равно (=) и не равно (≠), а *логическими операциями* – одноместная операция отрицание ($^{-}$), двухместные операции конъюнкция (&) и дизъюнкция (∨). Результатом операции отношения, а также операндом и результатом логической операции может быть либо истина (1), либо ложь (0). Операции дизъюнкция, конъюнкция и отрицание заданы табл. 5.2 – 5.4.

Таблица 5.2

a	$\overline{a}$
0	1
1	0

Таблица 5.3

a	b	$a \,\&\, b$
0	0	0
0	1	0
1	0	0
1	1	1

Таблица 5.4

a	b	$a \vee b$
0	0	0
0	1	1
1	0	1
1	1	1

Приведём основные законы, которым удовлетворяют заданные операции:

– *идемпотентности* дизъюнкции и конъюнкции

$$a \vee a = a, \quad a \,\&\, a = a;$$

– *коммутативности* дизъюнкции и конъюнкции

$$a \vee b = b \vee a, \quad a \,\&\, b = b \,\&\, a;$$

– *ассоциативности* дизъюнкции и конъюнкции

$$a \vee (b \vee c) = (a \vee b) \vee c, \quad a \,\&\, (b \,\&\, c) = (a \,\&\, b) \,\&\, c;$$

– *дистрибутивности* конъюнкции относительно дизъюнкции и дизъюнкции относительно конъюнкции

$$a \,\&\, (b \vee c) = a \,\&\, b \vee a \,\&\, c, \quad a \vee (b \,\&\, c) = (a \vee b) \,\&\, (a \vee c);$$

– *двойного отрицания*

$$\overline{\overline{a}} = a\,;$$

– *де-Моргана*

$$\overline{a} \vee \overline{b} = \overline{a \,\&\, b}\,, \quad \overline{a} \,\&\, \overline{b} = \overline{a \vee b}\,;$$

– *склеивания*

$$a \,\&\, b \vee a \,\&\, \overline{b} = a\,, \quad (a \vee b) \,\&\, (a \vee \overline{b}) = a\,;$$

– *поглощения*

$$a \vee a \,\&\, b = a, \quad a \,\&\, (a \vee b) = a;$$

– *действий с константами* 0 и 1

$$a \vee 0 = a, \quad a \,\&\, 0 = 0, \quad a \vee 1 = 1,$$

$$a \,\&\, 1 = a, \quad a \vee \overline{a} = 1\,, \quad a \,\&\, \overline{a} = 0\,.$$

При составлении логических выражений необходимо учитывать приоритет выполняемых операций: в первую очередь выполняется отрицание, затем конъюнкция и в последнюю очередь дизъюнкция. Для изменения порядка выполнения логических операций в выражениях следует использовать скобки.

Для получения требуемого логического выражения сначала, используя координаты x и y точки, операции отношения и конъюнкцию, необходимо составить логические выражения для логических переменных k, p и t. Эти переменные обозначают соответственно высказывания «точка с координатами (x, y) принадлежит кругу», «точка с координатами (x, y) принадлежит прямоугольнику» и «точка с координатами (x, y) принадлежит треугольнику». Каждая из переменных k, p и t или её отрицание называется *первичным термом*.

Затем следует получить конъюнкции первичных термов для каждого из заданных в табл. 5.1 множеств. При этом, если, например, заданное множество принадлежит кругу, в конъюнкцию следует включить саму переменную k, в противном случае – её отрицание $\overline{k}$. Аналогичным образом следует поступить с включением переменных p и t или их отрицаний $\overline{p}$ и $\overline{t}$. Каждая из полученных конъюнкций называется *конституентой*.

Наконец, записать дизъюнкцию конституент, которая и будет представлять собой СДНФ.

Количество первичных термов в представлении логического выражения дизъюнктивной нормальной формой называется *сложностью* представления.

Кроме задания логического выражения дизъюнкцией конституент его можно задать табличным способом и гиперкубом.

При табличном задании логического выражения строят прямоугольную таблицу, столбцам которой сопоставляют переменные и само логическое выражение, а в строках записывают всевозможные комбинации значений переменных и соответствующие им значения логического выражения.

При задании логического выражения с помощью гиперкуба строят неориентированный граф, каждой вершине которого взаимно однозначно соответствует двоичный набор значений переменных; вершины упорядочивают по ярусам: в i-й ярус входят вершины, которым соответствуют двоичные наборы, содержащие i единиц; вершины соединяют ребром, если соответствующие им наборы отличаются в одном и только одном разряде; вершины, соответствующие наборам значений переменных, при которых значение логического выражения равно 1, заштриховываются.

Минимальной ДНФ логического выражения называется ДНФ этого выражения, имеющая минимальную сложность.

Для получения минимальной ДНФ логического выражения может быть использован метод Квайна (импликантных таблиц) [3]. Этот метод заключается в последовательном выполнении двух этапов.

Этап 1. Выделение максимальных единичных интервалов.

Под *единичным интервалом I* понимается множество интервалов, на которых логическое выражение принимает значение 1, которое образует гиперкуб некоторой размерности. Мощность интервала равна степени числа 2: интервал размерности 2^0 – вершина, 2^1 – ребро, 2^2 – грань и т.д.

Интервал I_α называется *максимальным интервалом* логического выражения, если не найдётся другого интервала I_β этого выражения, содержащего в себе интервал I_α.

Конъюнкция, соответствующая максимальному единичному интервалу логического выражения, называется *простой импликантой* этого выражения.

Дизъюнкция простых импликант называется *сокращённой ДНФ* логического выражения.

Получением сокращённой ДНФ логического выражения заканчивается первый этап метода.

Тупиковой ДНФ логического выражения называется такая ДНФ этого выражения, которая при вычёркивании хотя бы одного первичного терма не определяет это логическое выражение. Минимальная ДНФ логического выражения является тупиковой.

Построение тупиковых ДНФ логического выражения сводится к покрытию столбцов строками в двумерной таблице. *Покрытием столбцов строками в двумерной таблице* называется такое множество строк, при котором для каждого столбца найдётся хотя бы одна строка, на пересечении с которой этот столбец имеет единицу, причём при вычёркивании хотя бы одного элемента из этого множества строк указанное свойство не выполняется.

Этап 2. Построение и покрытие таблицы Квайна.

Таблица Квайна – двумерная таблица, каждой строке которой взаимно однозначно соответствует максимальный единичный интервал, столбцу – конституента, а на пересечении i-й строки и j-го столбца ставится единица, если j-я конституента входит в i-й интервал, а в противном случае клетка (i, j) не заполняется или в ней ставится ноль.

Минимальная ДНФ выбирается из тупиковых ДНФ, соответствующих покрытиям таблицы Квайна. Покрытия таблицы Квайна определяются путём преобразования некоторой мультипликативно-аддитивной формы в аддитивно-мультипликативную.

Дальнейшая минимизация логического выражения иногда возможна при переходе из ДНФ в скобочную форму с применением закона дистрибутивности конъюнкции относительно дизъюнкции.

Для минимального логического выражения строится структурная вычислительная схема с использованием блоков, реализующих операции отрицание, конъюнкция и дизъюнкция, которые представлены на рис. 5.2, 5.3 и 5.4 соответственно.

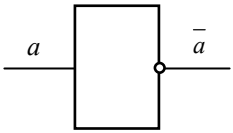


Рис. 5.2

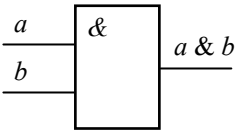


Рис. 5.3

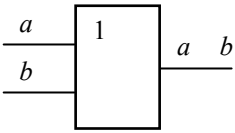
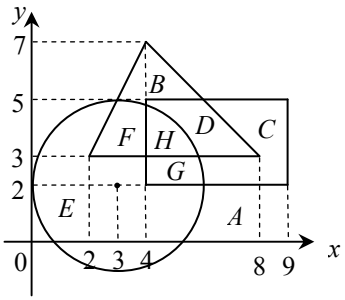


Рис. 5.4

Примеры выполнения задания

Пример № 1

З*адание*. На плоскости *x*0*y* заданы круг, прямоугольник и треугольник. Плоскость разбита линиями, ограничивающими эти фигуры, на непересекающиеся множества точек *A*, *B*, *C*, *D*, *E*, *F*, *G* и *H*. Составить совершенную дизъюнктивную нормальную форму (ДНФ) логического выражения, истинного только для точек объединения заданных множеств *A*, *D*, *E*, *G* и *H*, минимизировать её и построить соответствующую структурную вычислительную схему.



Решение.

Введём логические переменные *k*, *p* и *t*, которые будут принимать значение истина только в том случае, когда точка с координатами (*x*, *y*) принадлежит кругу, прямоугольнику и треугольнику соответственно.

Для получения требуемого логического выражения сначала, используя координаты *x* и *y* точки, а также операции отношения и конъюнкцию, составим логические выражения для введённых логических переменных.

Логическим выражением для *k* будет отношение $(x-3)^2 + (y-2)^2 \le 3^2$, определяющее заданный круг с центром в точке (3, 2) и радиусом 3.

Переменную *p* будет представлять конъюнкция ($x \ge 4$) & ($x \le 9$) & & ($y \ge 2$) & ($y \le 5$), соответствующая заданному прямоугольнику с коор-

динатами противоположных углов (4, 2) и (9, 5), который получается в результате пересечения четырёх полуплоскостей $x \ge 4$, $x \le 9$, $y \ge 2$ и $y \le 5$.

При построении логического выражения для t сначала найдём уравнения трёх прямых, проходящих через пары вершин треугольника. Через вершины с координатами (2, 3) и (4, 7) проходит прямая $y = 2x - 1$, через вершины (4, 7) и (8, 3) – прямая $y = 11 - x$, и через вершины (8, 3) и (2, 3) – прямая $y = 3$. Заданный треугольник образуется в результате пересечения трех полуплоскостей: $t = (y \le 2x - 1)\ \&\ (y \le 11 - x)\ \&\ (y \ge 3)$.

Получим конъюнкции первичных термов для каждого из заданных множеств A, D, E, G и H. В результате множеству A будет соответствовать конъюнкция $\bar{k}\ \&\ \bar{p}\ \&\ \bar{t}$, множеству D – конъюнкция $\bar{k}\ \&\ p\ \&\ t$, множеству $E - k\ \&\ \bar{p}\ \&\ \bar{t}$, множеству $G - k\ \&\ p\ \&\ \bar{t}$ и, наконец, множеству $H - k\ \&\ p\ \&\ t$.

Каждая из полученных конъюнкций является конституентой, дизъюнкция которых и будет представлять искомую совершенную ДНФ:

$$\bar{k}\ \&\ \bar{p}\ \&\ \bar{t} \vee \bar{k}\ \&\ p\ \&\ t \vee k\ \&\ \bar{p}\ \&\ \bar{t} \vee k\ \&\ p\ \&\ \bar{t} \vee k\ \&\ p\ \&\ t.$$

Без символа операции конъюнкция & совершенная ДНФ имеет вид

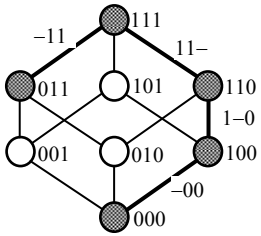
$$\bar{k}\ \bar{p}\ \bar{t} \vee \bar{k}\ p\ t \vee k\ \bar{p}\ \bar{t} \vee k\ p\ \bar{t} \vee k\ p\ t.$$

Заметим, что сложность этой ДНФ равна 15.

Представим логическое выражение табличным способом, указав в качестве значения выражения 1 тогда и только тогда, когда соответствующее множество задано, т.е. в строках A, D, E, G и H.

Множество	Значения переменных			Значение выражения
	k	p	t	
A	0	0	0	1
B	0	0	1	0
C	0	1	0	0
D	0	1	1	1
E	1	0	0	1
F	1	0	1	0
G	1	1	0	1
H	1	1	1	1

Гиперкуб логического выражения будет иметь следующий вид.



Выполним минимизацию составленного логического выражения в классе ДНФ.

Этап 1. Выделение максимальных единичных интервалов.

Запишем множество единичных интервалов для рассматриваемого примера $I = \{000, 100, 110, 111, 011, –00, 1–0, 11–, –11\}$. Здесь и далее «–» означает, что переменная, соответствующая этому разряду, в конъюнкции отсутствует, т.е. по этой переменной после дизъюнкции соответствующих конъюнкций произошло склеивание. Так, например, интервал –00, соответствующий множеству конституент {000, 100}, получается в результате преобразования $\bar{k}\,\bar{p}\,\bar{t} \vee k\,\bar{p}\,\bar{t} = \bar{p}\,\bar{t}$. Первые пять из выписанных выше единичных интервалов обозначают вершины гиперкуба, а остальные – его рёбра. Соответствующие вершины гиперкуба на рисунке заштрихованы, а рёбра отмечены толстой линией.

Составим множество максимальных интервалов $I_{\max} = \{–00, 1–0, 11–, –11\}$. Это множество содержит четыре ребра, поскольку каждая единичная вершина принадлежит, по крайней мере, одному единичному ребру, и ни одно единичное ребро не содержится ни в какой единичной грани.

Запишем сокращённую дизъюнктивную нормальную форму рассматриваемого логического выражения:

$$\bar{p}\,\bar{t} \vee k\,\bar{t} \vee k\,p \vee p\,t\,.$$

Этап 2. Построение и покрытие таблицы Квайна.

Построим прямоугольную таблицу Квайна, каждой строке которой взаимно-однозначно сопоставим максимальные единичные интервалы – 00, 1–0, 11– и –11, а столбцам – конституенты 000, 100, 110, 011 и 111.

	Максимальные единичные интервалы	Конституента				
		000	100	110	011	111
a	–00	1	1	0	0	0
b	1–0	0	1	1	0	0
c	11–	0	0	1	0	1
d	–11	0	0	0	1	1

Для получения покрытий столбцов строками обозначим строки таблицы Квайна соответственно буквами *a*, *b*, *c* и *d*. Для каждого столбца запишем дизъюнкцию строк, покрывающих этот столбец, соединим полученные дизъюнкции знаком конъюнкции и преобразуем полученное произведение сумм в сумму произведений:

$$a \,\&\, (a \vee b) \,\&\, (b \vee c) \,\&\, d \,\&\, (c \vee d) = a \,\&\, (b \vee c) \,\&\, d =$$
$$= (a \,\&\, b \vee a \,\&\, c) \,\&\, d = a \,\&\, b \,\&\, d \vee a \,\&\, c \,\&\, d.$$

Полученные конъюнкции a & b & d и a & c & d обозначают покрытия {–00, 1–0, –11} и {–00, 11–, –11}, соответствующие тупиковым ДНФ $\bar{p}\bar{t} \vee k\bar{t} \vee pt$ и $\bar{p}\bar{t} \vee kp \vee pt$. В качестве минимальной ДНФ можно выбрать любую из этих двух тупиковых ДНФ, поскольку сложность каждой из них равна 6. Выберем, например, первую:

$$\bar{p}\bar{t} \vee k\bar{t} \vee pt .$$

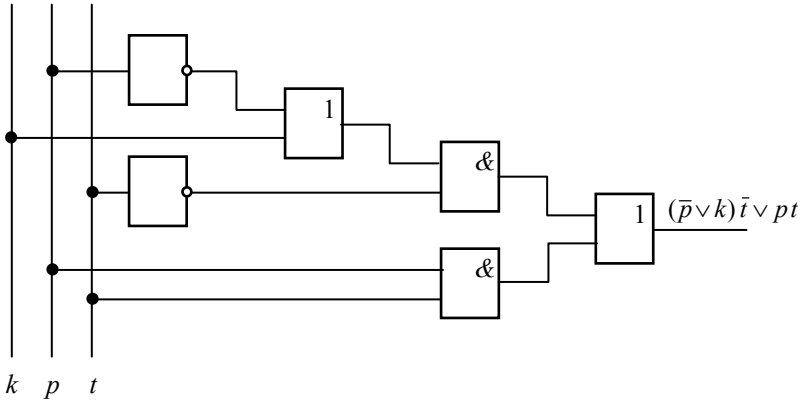
Заметим, что в результате минимизации методом Квайна сложность представления логического выражения в классе ДНФ понизилась с 15 до 6.

Дальнейшая минимизация логического выражения в данном случае возможна при переходе из ДНФ в скобочную форму с применением закона дистрибутивности конъюнкции относительно дизъюнкции:

$$\bar{p}\bar{t} \vee k\bar{t} \vee pt = (\bar{p} \vee k)\bar{t} \vee pt .$$

Заметим, что в результате перехода к скобочной форме сложность представления логического выражения понизилась с 6 до 5.

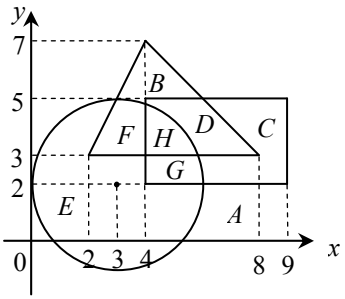
Для скобочной формы логического выражения построим структурную вычислительную схему с использованием блоков, реализующих операции отрицание, конъюнкция и дизъюнкция.



Ответ: $(\bar{p} \vee k)\bar{t} \vee pt$.

Пример № 2

З*адание*. На плоскости *x*0*y* заданы круг, прямоугольник и треугольник. Плоскость разбита линиями, ограничивающими эти фигуры, на непересекающиеся множества точек *A*, *B*, *C*, *D*, *E*, *F*, *G* и *H*. Составить совершенную дизъюнктивную нормальную форму (ДНФ) логического выражения, истинного только для точек объединения заданных множеств *A*, *C*, *D*, *E* и *G*, минимизировать её и построить соответствующую структурную вычислительную схему.



Решение.

Введём логические переменные k, p и t, которые будут принимать значение истина только в том случае, когда точка с координатами (x, y) принадлежит кругу, прямоугольнику и треугольнику соответственно.

Логические выражения для введённых логических переменных k, p и t будут такими же, как и в первом примере.

Получим конъюнкции первичных термов для заданных множеств A, C, D, E и G. В результате множеству A будет соответствовать конъюнкция $\bar{k} \,\&\, \bar{p} \,\&\, \bar{t}$, множеству C – конъюнкция $\bar{k} \,\&\, p \,\&\, \bar{t}$; множеству D – $\bar{k} \,\&\, p \,\&\, t$; множеству E – $k \,\&\, \bar{p} \,\&\, \bar{t}$ и, наконец, множеству G – $k \,\&\, p \,\&\, \bar{t}$.

Каждая из полученных конъюнкций является конституентой, дизъюнкция которых и будет представлять искомую совершенную ДНФ:

$$\bar{k} \,\&\, \bar{p} \,\&\, \bar{t} \vee \bar{k} \,\&\, p \,\&\, \bar{t} \vee \bar{k} \,\&\, p \,\&\, t \vee k \,\&\, \bar{p} \,\&\, \bar{t} \vee k \,\&\, p \,\&\, \bar{t}.$$

Без символа операции конъюнкция & совершенная ДНФ имеет вид

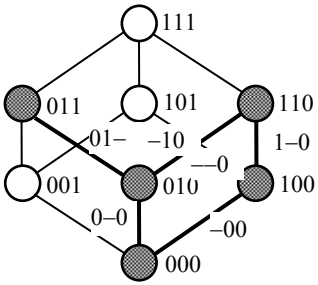
$$\bar{k}\ \bar{p}\ \bar{t} \vee \bar{k}\ p\ \bar{t} \vee \bar{k}\ p\ t \vee k\ \bar{p}\ \bar{t} \vee k\ p\ \bar{t}.$$

Заметим, что сложность представления этой ДНФ равна 15.

Представим логическое выражение табличным способом, указав при этом в качестве значения выражения величину 1 тогда и только тогда, когда соответствующее множество задано, т.е. в строках A, C, D, E и G.

Множество	Значения переменных			Значение выражения
	k	p	t	
A	0	0	0	1
B	0	0	1	0
C	0	1	0	1
D	0	1	1	1
E	1	0	0	1
F	1	0	1	0
G	1	1	0	1
H	1	1	1	0

Гиперкуб логического выражения изображён на следующем рисунке.



Выполним минимизацию составленного логического выражения в классе ДНФ.

Этап 1. Выделение максимальных единичных интервалов.

Запишем множество единичных интервалов для рассматриваемого примера I = {000, 010, 100, 011, 110, 0–0, –00, 01–, –10, 1–0, ––0}. Здесь и далее «–» означает, что переменная, соответствующая этому разряду, в конъюнкции отсутствует, т.е. по этой переменной после дизъюнкции соответствующих конъюнкций произошло склеивание. Первые пять из выписанных выше единичных интервалов обозначают вершины гиперкуба, следующие пять – рёбра, а последний – грань.

Составим множество максимальных интервалов I_{max} = {01–, ––0}. Это множество содержит ребро 01– и грань ––0, поскольку каждая единичная вершина принадлежит, по крайней мере, одному единичному ребру, а рёбра 0–0, –00, –10 и 1–0 содержатся в грани ––0.

Запишем сокращённую ДНФ логического выражения:

$$\bar{k}\, p \vee \bar{t}\,.$$

Этап 2. Построение и покрытие таблицы Квайна.

Таблица Квайна для рассматриваемого примера будет иметь следующий вид.

	Максимальные единичные интервалы	Конституента				
		000	010	100	011	110
a	01–	0	1	0	1	0
b	––0	1	1	1	0	1

Для получения покрытий столбцов строками обозначим строки таблицы Квайна буквами *a* и *b*. Для каждого столбца запишем дизъюнкцию строк, покрывающих этот столбец, соединим полученные дизъюнкции знаком конъюнкции и преобразуем полученное произведение сумм в сумму произведений:

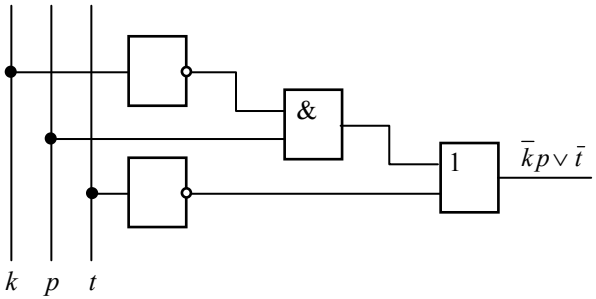
$$b \,\&\, (a \vee b) \,\&\, b \,\&\, a \,\&\, b = a \,\&\, (a \vee b) \,\&\, b \,\&\, b \,\&\, b = a \,\&\, b.$$

Полученная сумма из одного слагаемого a & b обозначает покрытие {01–, ––0} и соответствует единственной тупиковой ДНФ $\bar{k}\, p \vee \bar{t}$, которую и следует считать минимальной.

Заметим, что в результате минимизации сложность представления логического выражения в классе ДНФ уменьшилась с 15 до 3.

Дальнейшая минимизация логического выражения путём перехода из ДНФ в скобочную форму в данном случае не возможна.

Для минимальной ДНФ логического выражения построим структурную вычислительную схему с использованием блоков, реализующих операции отрицания, конъюнкции и дизъюнкции.



Ответ: $\bar{k}\, p \vee \bar{t}$.

Лабораторная работа № 6

МАШИНА ТЬЮРИНГА

Цель: знакомство с понятием машины Тьюринга и приобретение навыков конструирования машин Тьюринга.

Задание. Сконструировать машину Тьюринга, которая реализует заданную в табл. 6.1 метаграмму.

Основные положения

Понятие алгоритма является одним из основных понятий современной информатики. Термин *алгоритм* (алгорифм) происходит от имени среднеазиатского учёного IX в. аль-Хорезми, который разработал правила выполнения четырёх арифметических действий в десятичной системе счисления.

Вплоть до 30-х гг. прошлого столетия понятие алгоритма носило интуитивный характер и имело скорее методологическое, чем математическое значение. Общей теории алгоритмов не существовало, а под алгоритмом понимали *конечную совокупность точно сформулированных правил*, *которые позволяли решать те или иные классы задач*.

Таблица 6.1

Вариант	Метаграмма
1	ЗУБ → КУБ → КУМ → КОМ → КОТ
2	ДЕНЬ → СЕНЬ → СЕЛЬ → СОЛЬ
3	ТЕСТО → МЕСТО → МЕСТЬ
4	ДЫМ → ДОМ → СОМ → СОР → БОР
5	ДЕНЬ → ПЕНЬ → ПЕНА → ВЕНА
6	МАСТЬ → ПАСТЬ → ПАСТА
7	БАР → ПАР → ПИР → ПИК → ЛИК
8	ВИЗА → ВАЗА → ФАЗА → ФАРА
9	КАСТА → КАСКА → КАЧКА
10	СУП → СУК → СОК → СОМ → РОМ
11	КАРА → КОРА → КОЖА → ЛОЖА
12	ПАЧКА → ПАЛКА → БАЛКА
13	РАК → РОК → СОК → СОМ → КОМ
14	ЛОЖЬ → РОЖЬ → РОЛЬ → НОЛЬ
15	ТЕСТО → МЕСТО → МЕСТЬ
16	ЛОМ → КОМ → ДОМ → ДОК → КОК
17	ЛОЖА → ЛУЖА → ЛУКА → МУКА
18	МАСТЬ → ПАСТЬ → ПАСТА
19	ЩИТ → КИТ → КОТ → ПОТ → ПОЛ
20	МРАК → БРАК → БРУС → ТРУС
21	КАСТА → КАРТА → ПАРТА
22	МИГ → МИР → ПИР → ПАР → ПАС
23	ТРОН → УРОН → УРОК → СРОК
24	ВОРОНА → КОРОНА → КОРОВА
25	ЧАС → БАС → БАР → БОР → ХОР
26	РЕКА → РУКА → ЛУКА → ЛУЖА
27	КАПОТ → КАПОР → НАПОР
28	ХОД → ГОД → РОД → РОВ → ЛОВ
29	СТОК → СТОН → СТАН → СТАЯ
30	БУРАН → БАРАН → БАРОН

В середине 30-х гг. прошлого века было получено формализованное понятие алгоритма, позволяющее доказывать алгоритмическую неразрешимость некоторых задач.

Так в 1936–1937 гг. независимо друг от друга дали определение понятия алгоритма американский и английский математики Э. Пост (1897 – 1954) и А. Тьюринг (1912 – 1954). Их основной мыслью было то, что алгоритмические процессы, когда-либо описываемые математиками, совершает подходяще устроенная «машина». Абстрактные машины, введённые Постом и Тьюрингом, отличались не очень существенно и в дальнейшем стали именоваться машинами Тьюринга.

Машиной Тьюринга называется специальный абстрактный автомат, имитирующий алгоритмические процессы [4]. Она состоит из информационной ленты (ИЛ) и управляющей головки (УГ).

Информационная лента представляет собой бесконечную ленту, разделённую на ячейки. В каждую ячейку ленты помещается один символ из некоторого конечного множества символов $A = \{s_0, s_1, \ldots, s_n\}$, называемого *внешним алфавитом* машины. Будем полагать, что символ s_0 алфавита A равен 0 (нулю). Если в ячейке ленты записан символ 0 (ноль), то она считается пустой ячейкой.

Управляющая головка всегда находится в одном из некоторого конечного множества состояний $Q = \{q_0, q_1, \ldots, q_m\}$, называемого *внутренним алфавитом* машины. Элемент q_1 из множества Q обозначает начальное, а символ q_0 – конечное состояние (машина выключена). УГ может считывать символ из ячейки и записывать символ в ячейку, которую «видит». Кроме того, она может перемещаться вдоль информационной ленты на одну ячейку влево (Л), на одну ячейку вправо (П) или оставаться на месте (Н). Символы Л, П и Н образуют *алфавит перемещений D*.

Машина Тьюринга работает в дискретные моменты времени, называемые тактами. В каждый такт работы УГ совершает следующие действия: 1) считывает символ s_i, который она «видит»; 2) в соответствии со считанным символом s_i и своим состоянием q_j записывает символ s_k в эту ячейку; 3) переходит в следующее состояние q_p; 4) перемещается или не перемещается вдоль ленты (символ $d_l \in D$).

Работу машины Тьюринга можно задать с помощью функциональной таблицы T, строки которой определяют символы алфавита Q, а столбцы – символы алфавита A. В ячейку таблицы T, находящуюся на пересечении строки q_j и столбца s_i, в соответствии со сказанным выше записывают упорядоченную тройку $s_k\, q_p\, d_l$.

Совокупность содержания информационной ленты, положения и состояния управляющей головки называют *конфигурацией* машины Тьюринга. Описать всю работу машины Тьюринга можно последовательностью её конфигураций от пуска до останова.

Пример выполнения задания

Задание. Сконструировать машину Тьюринга, которая реализует заданную метаграмму ГРИБ → ГРИМ → ГРОМ → ГНОМ.

Решение.

Во внешний алфавит A машины Тьюринга войдут обязательный символ 0 (ноль), обозначающий пустую ячейку информационной ленты, а также буквы Б, Г, И, М, Н, О и Р, используемые в словах заданной метаграммы ГРИБ → ГРИМ → ГРОМ → ГНОМ, т.е. алфавит A = {0, Б, Г, И, М, Н, О, Р}.

Первый переход ГРИБ → ГРИМ заданной метаграммы определяется (первой) заменой Б → М, второй переход ГРИМ → ГРОМ – второй заменой И → О, последний переход ГРОМ → ГНОМ – третьей заменой Р → Н.

Состояниями q_1, q_2 и q_3 управляющей головки будем считать *состояния ожидания соответственно первой, второй и третьей замены при просмотре слова в направлении слева-направо*. Кроме того, введём состояния q_4, q_5 и q_6, обозначающие *состояния возврата к началу слова после сделанной соответственно первой, второй и третьей замены*. Следовательно, внутренний алфавит Q машины Тьюринга с учётом конечного состояния q_0 будет $Q = \{q_0, q_1, q_2, q_3, q_4, q_5, q_6\}$.

Алфавит перемещений D машины постоянен, т.е. D = {Л, П, Н}.

Пусть перед включением машины УГ находится в начальном состоянии q_1 и «видит» первую букву исходного слова ГРИБ, т.е. букву Г. Тогда для реализации заданной метаграммы ГРИБ → ГРИМ → ГРОМ → ГНОМ управляющая головка машины должна действовать следующим образом.

Если УГ находится в состоянии q_1 ожидания первой замены и видит не подлежащую первой замене букву Г, И или Р, она должна записать в обозреваемую ячейку такую же букву, перейти в такое же состояние q_1 ожидания первой замены и переместиться на одну ячейку вправо к следующей букве слова. Если же УГ, находясь в состоянии q_1, видит подлежащую первой замене букву Б, она должна записать на её место букву М, перейти в состояние q_4 возврата к началу слова после сделанной первой замены и переместиться на одну ячейку влево, сделав тем самым один шаг на пути к началу слова.

Если УГ находится в состоянии q_2 ожидания второй замены и видит не подлежащую второй замене букву Г или Р, она должна записать в обозреваемую ячейку такую же букву, перейти в такое же состояние q_2 ожидания второй замены и переместиться на одну ячейку вправо к следующей букве слова. Если же УГ, находясь в состоянии q_2, видит подлежащую второй замене букву И, она должна записать на её место букву О, перейти в состояние q_5 возврата к началу слова после сделанной второй замены и переместиться на одну ячейку влево, сделав тем самым один шаг на пути к началу слова.

Если УГ находится в состоянии q_3 ожидания третьей замены и видит не подлежащую третьей замене букву Г, она должна записать в обозреваемую ячейку такую же букву, перейти в такое же состояние q_3 ожидания третьей замены и переместиться на одну ячейку вправо к следующей букве слова. Если же УГ, находясь в состоянии q_3, видит подлежащую третьей замене букву Р, она должна записать на её место букву Н, перейти в состояние q_6 возврата к началу слова после сделанной третьей замены и переместиться на одну ячейку влево, сделав тем самым один шаг на пути к началу слова.

Если УГ находится в состоянии q_4 возврата к началу слова после первой замены и видит букву Г, И или Р, она должна записать в обозреваемую ячейку такую же букву, перейти в такое же состояние q_4 возврата к началу слова после первой замены и переместиться на одну ячейку влево, продолжив тем самым движение к началу слова. Если же УГ, находясь в состоянии q_4, видит пустой символ 0 (ноль), она сделала дополнительный шаг влево за левую границу слова и должна записать на его место такой же пустой символ, перейти в состояние q_2 ожидания второй замены и переместиться на одну ячейку вправо с целью увидеть первую букву слова.

Если УГ находится в состоянии q_5 возврата к началу слова после второй замены и видит букву Г или Р, она должна записать в обозреваемую ячейку такую же букву, перейти в такое же состояние q_5 возврата к началу слова после второй замены и переместиться на одну ячейку влево, продолжив тем самым движение к началу слова. Если же УГ, находясь в состоянии q_5 видит пустой символ 0 (ноль), она сделала дополнительный шаг влево за левую границу слова и должна записать на его место такой же пустой символ, перейти в состояние q_3 ожидания третьей замены и переместиться на одну ячейку вправо с целью увидеть первую букву слова.

Если УГ находится в состоянии q_6 возврата к началу слова после третьей (последней) замены и видит букву Г, она должна записать в обозреваемую ячейку такую же букву, перейти в такое же состояние q_6 возврата к началу слова после третьей замены и переместиться на одну ячейку влево, продолжив тем самым движение к началу слова. Если же УГ, находясь в состоянии q_6, видит пустой символ 0 (ноль), она сделала дополнительный шаг влево за левую границу слова и должна записать на его место такой же пустой символ, перейти в конечное состояние q_0 и переместиться на одну ячейку вправо с целью остановиться после выключения на первой букве конечного слова метаграммы.

Работу сконструированной машины Тьюринга можно задать компактно и формально с помощью функциональной таблицы T:

Q \ A	0 (ноль)	Б	Г	И	М	Н	О	Р
q_1		М q_4 Л	Г q_1 П	И q_1 П				Р q_1 П
q_2			Г q_2 П	О q_5 Л				Р q_2 П
q_3			Г q_3 П					Н q_6 Л
q_4	0 q_2 П		Г q_4 Л	И q_4 Л				Р q_4 Л
q_5	0 q_3 П		Г q_5 Л					Р q_5 Л
q_6	0 q_0 П		Г q_6 Л					

Опишем всю работу машины Тьюринга от пуска до останова последовательностью её конфигураций:

Пуск

…	0	0	Г	Р	И	Б	0	0	…

q_1

Такт 1

…	0	0	Г	Р	И	Б	0	0	…

q_1

Такт 2

…	0	0	Г	Р	И	Б	0	0	…

q_1

Такт 3

…	0	0	Г	Р	И	Б	0	0	…

q_1

Такт 4

…	0	0	Г	Р	И	М	0	0	…

q_4

Такт 5

…	0	0	Г	Р	И	М	0	0	…

q_4

Такт 6

…	0	0	Г	Р	И	М	0	0	…

q_4

Такт 7

…	0	0	Г	Р	И	М	0	0	…

q_4

Такт 8

…	0	0	Г	Р	И	М	0	0	…

q_2

Такт 9

…	0	0	Г	Р	И	М	0	0	…

q_2

Такт 10

…	0	0	Г	Р	И	М	0	0	…

q_2

Такт 11

…	0	0	Г	Р	О	М	0	0	…

q_5

Такт 12

…	0	0	Г	Р	О	М	0	0	…

q_5

Такт 13

…	0	0	Г	Р	О	М	0	0	…

q_5

Такт 14

…	0	0	Г	Р	О	М	0	0	…

q_3

Такт 15

…	0	0	Г	Р	О	М	0	0	…

q_3

Такт 16

…	0	0	Г	Н	О	М	0	0	…

q_6

Такт 17

…	0	0	Г	Н	О	М	0	0	…

q_6

Такт 18

…	0	0	Г	Н	О	М	0	0	…

q_0

Останов

Лабораторная работа № 7

НОРМАЛЬНЫЕ АЛГОРИТМЫ МАРКОВА

Цель: знакомство с понятием нормальных алгоритмов Маркова и приобретение навыков их конструирования.

Задание.

Сконструировать нормальный алгоритм Маркова, который реализует заданную в табл. 7.1 метаграмму.

Таблица 7.1

Вариант	Метаграмма
1	МУХА → МУРА → ТУРА → ТАРА → ПАРА → ПАРК → → ПАУК → ПАУТ → ПЛУТ → ПЛОТ → СЛОТ → СЛОН
2	ДЕНЬ → ПЕНЬ → ПЕНА → ВЕНА → ВИНА → ВИЗА → ВАЗА → ФАЗА → ФАРА → КАРА → КОРА → КОЖА → ЛОЖА → → ЛОЖЬ → РОЖЬ → РОЛЬ → НОЛЬ → НОЧЬ
3	МРАК → БРАК → БРУС → ТРУС → ТРОС → ТРОН → → УРОН → УРОК → СРОК → СТОК → СТОН → СТАН → → СТАЯ → СВАЯ → СВАТ → СВЕТ
4	ГОРА → ПОРА → ПОЛА → ПОЛО → СОЛО → СЕЛО → СЕНО → СЕНИ → САНИ
5	БУРАК → БУРАН → БАРАН → БАРОН → БАТОН → ЗАТОН → ЗАГОН → ВАГОН
6	ТЕСТО → МЕСТО → МЕСТЬ → МАСТЬ → ПАСТЬ → ПАСТА → КАСТА → КАСКА → КАЧКА → ПАЧКА → ПАЛКА → → БАЛКА → БЕЛКА → БУЛКА
7	РЕКА → РУКА → ЛУКА → ЛУЖА → ЛОЖА → ЛОЗА → → ПОЗА → ПОРА → ГОРА → ГОРЕ → МОРЕ
8	ЛИПА → ЛАПА → ПАПА → ПАРА → КАРА → КОРА → → КОСА → РОСА → РОЩА
9	РЕЙС → РЕПС → РЕПА → РЕКА → РУКА → ЛУКА → ЛУЖА → ЛОЖА → КОЖА → КОРА → КАРА → ФАРА → ФАЗА → ВАЗА → ВИЗА → ВИНА → ШИНА
10	НОГА → НОТА → РОТА → РОЗА → ЛОЗА → ЛУЗА → МУЗА → МУКА → РУКА
11	ЛУНА → ЛУПА → ЛАПА → ЛАМА → РАМА → РАСА → → РОСА → КОСА → КОРА → ГОРА → ГОРЕ → МОРЕ → → МОРС → МАРС
12	ТЕСТО → МЕСТО → МЕСТЬ → МАСТЬ → ПАСТЬ → ПАСТА → КАСТА → КАРТА → ПАРТА → ПАРКА → БАРКА → → БУРКА → БУЛКА
13	ВОЛК → ПОЛК → ПОЛА → ПОРА → ПАРА → ПАВА → → ЛАВА → ЛАВР → ЛИВР → ЛИТР → ТИТР → ТИГР
14	БАРС → ФАРС → ФАРА → ПАРА → ПАПА → ЛАПА → → ЛИПА → ЛИСА

Продолжение табл. 7.1

Вариант	Метаграмма
15	КОЗА → РОЗА → РИЗА → ВИЗА → ВИНА → ВЕНА → ПЕНА → ПЕНЬ → ДЕНЬ → ЛЕНЬ → ЛАНЬ
16	ПАУК → ПАРК → ПАРА → КАРА → КУРА → КУПА → → ЛУПА → ЛУКА → МУКА → МУХА
17	КОШКА → МОШКА → МИШКА → МЫШКА
18	МУХА → МУРА → ТУРА → ТАРА → КАТА → КАРЕ → → КАФЕ → КАФР → КАЮР → КАЮК → КРЮК → КРОК → → СРОК → СТОК → СТОН → СЛОН
19	ДУША → СУША → СУШЬ → СУТЬ → СЕТЬ → СЕНЬ → → СЕНО → СЕЛО → ТЕЛО
20	ЛИСА → ЛИПА → ЛАПА → ПАПА → ПАРА → ПОРА → → НОРА
21	ОЛОВО → СЛОВО → СЛАВА → СЛЕВА → СЛОВА → → СЛИВА
22	НАГАН → КАГАН → КАГАЛ → КАГАТ
23	МАМА → МАРА → МАРЬ → ГАРЬ → ГАТЬ → МАТЬ
24	ЛОМ → КОМ → ДОМ → ДОК → КОК → КОЛ
25	СУМА → КУМА → КАМА → РАМА → МАМА → МАРА → → МАРЬ → ЛАРЬ → ЛАНЬ
26	ЛИНЬ → ЛУНЬ → ЛУНА → ЛУКА → ЛУЗА → ЛОЗА → КОЗА → КОРА → КОРЖ → МОРЖ
27	МИГ → МАГ → МАЙ → ЧАЙ → ЧАС → ЧАД → ГАД → → ГОД → КОД → КОК → КЕК → ВЕК → БЕК → БОК → → БОА → БРА → ЭРА
28	СИЛА → ПИЛА → ПОЛА → ПОРА → КОРА → КОЖА → → ЛОЖА → ЛУЖА → ЛУПА → ЛИПА → ЛИСА
29	ВИЛКА → ВАЛКА → ПАЛКА → ПОЛКА → ПОРКА → → НОРКА → КОРКА → КОШКА → КАШКА → КАСКА → → МАСКА → МИСКА
30	ПЕНЬ → ПЕНА → ВЕНА → ВИНА → ВИЗА → ВАЗА → → ФАЗА → ФАРА → КАРА → КОРА → КОЖА → ЛОЖА → → ЛОЖЬ → РОЖЬ

Основные положения

Нормальные алгоритмы Маркова относятся к алгоритмическим системам, основанным на соответствии между словами в абстрактном алфавите, и включают в себя объекты двух типов: распознаватели вхождения (РВ) и операторы подстановки (ОП) [4].

Распознаватель вхождения проверяет условие – имеет ли место вхождение рассматриваемого слова p_1 в качестве подслова некоторого заданного слова q.

Оператор подстановки заменяет первое слева вхождение слова p_1 в слово q на некоторое заданное слово p_2. Оператор подстановки задаётся обычно в виде двух слов, соединённых стрелкой $p_1 \rightarrow p_2$.

Для представления нормальных алгоритмов Маркова используют граф-схемы или упорядоченное множество подстановок.

Граф-схема нормального алгоритма Маркова представляет собой конечное множество вершин, соединённых между собой дугами. Каждой вершине граф-схемы, кроме особых вершин, называемых входом и выходом, сопоставляется какой-либо РВ или ОП. Все вершины, соответствующие распознавателям, упорядочиваются путём их нумерации от 1 до n. Дуги, исходящие из вершин, соответствующих операторам подстановки, подсоединяются либо к вершине, соответствующей первому распознавателю, либо к выходной вершине. В первом случае подстановка называется обычной, во втором – заключительной. Входная вершина соединяется дугой с первым распознавателем.

В *упорядоченном множестве подстановок* обычные подстановки записываются в виде двух слов, соединённых стрелкой ($p_1 \rightarrow p_2$), а в заключительных используется стрелка с точкой ($p_1 \rightarrow \bullet\ p_2$). Процесс выполнения подстановок заканчивается лишь тогда, когда (первый раз) выполнена какая-либо заключительная подстановка.

В теории алгоритмов строго доказано, что по своим возможностям преобразования нормальные алгоритмы Маркова эквивалентны машине Тьюринга и другим моделям, уточняющим понятие алгоритма.

Пример выполнения задания

Задание. Сконструировать нормальный алгоритм Маркова, который реализует заданную метаграмму МАСКА → КАСКА → КАШКА → → КОШКА → КОРКА → НОРКА → ПОРКА → ПОЛКА.

Решение.

На основании заданной метаграммы можно судить, что создаваемый для её реализации нормальный алгоритм Маркова будет основан на соответствии между словами в русском алфавите.

Для осуществления первого перехода МАСКА → КАСКА заданной метаграммы необходимо в слове МАСКА обнаружить вхождение буквы М ($РВ_1$) и выполнить обычную подстановку М → К ($ОП_1$). Повторное распознание вхождения буквы М (теперь в слово КАСКА) не даст положительного результата и управление выполнением алгоритма будет передано на следующий распознаватель ($РВ_2$).

Второй переход КАСКА → КАШКА должен обеспечить срабатывание распознавателя вхождения буквы С ($РВ_2$) и выполнение обычной подстановки С → Ш ($ОП_2$). Дальнейшее последовательное распознание вхождений букв М и С (в слово КАШКА) не даст положительного результата и управление выполнением алгоритма будет передано на распознаватель вхождения $РВ_3$.

Заметим, что третий переход КАШКА → КОШКА должен быть связан с заменой только первой слева буквы А на О. Поэтому распознавателю вхождения $РВ_3$ необходимо сработать, например, на вхождение подслова АШ и выполнить обычную подстановку АШ → ОШ ($ОП_3$). Тогда последовательное распознание вхождений подслов М, С и АШ (в слово КОШКА) не даст положительного результата и управление выполнением алгоритма будет передано на распознаватель вхождения $РВ_4$.

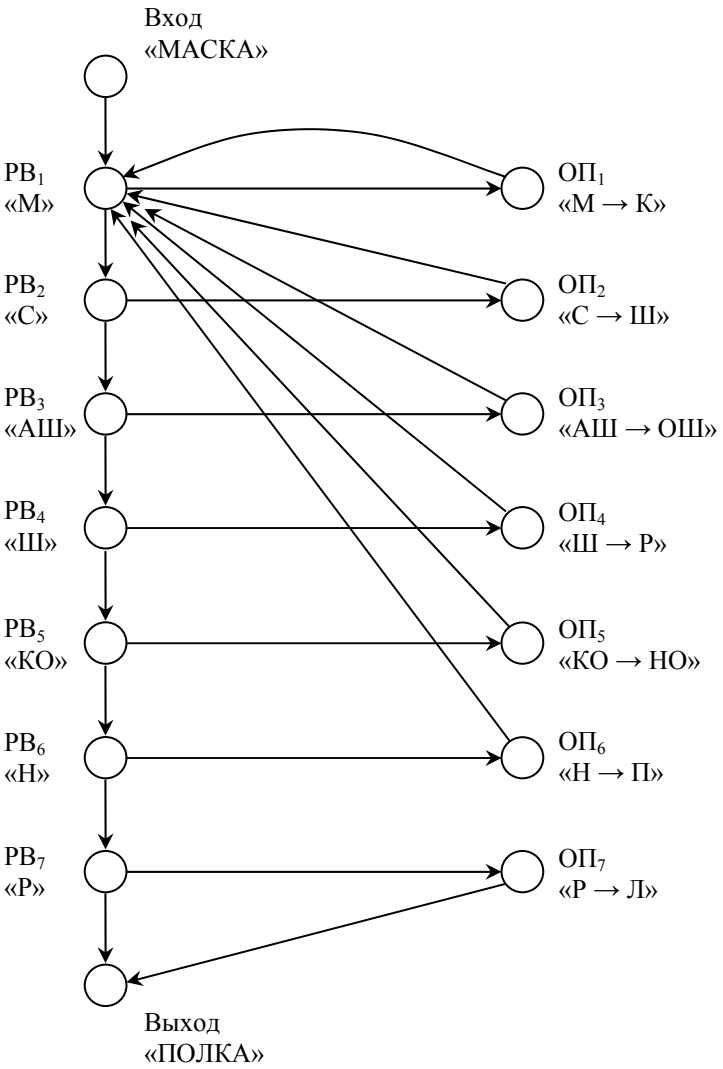
Для осуществления четвёртого перехода КОШКА → КОРКА должен сработать распознаватель буквы Ш ($РВ_4$) и выполнена обычная подстановка Ш → Р ($ОП_4$). Последовательное распознание вхождений подслов М, С, АШ и Ш в слово КОРКА не даст положительного результата и управление выполнением алгоритма будет передано на распознаватель $РВ_5$.

В пятом переходе КОРКА → НОРКА необходимо заменить только первую слева букву К на Н. Поэтому распознавателю вхождения $РВ_5$ требуется сработать, например, на вхождение подслова КО и выполнить обычную подстановку КО → НО ($ОП_5$). Тогда последовательное распознание вхождений подслов М, С, АШ, Ш и КО в слово НОРКА не даст положительного результата и управление выполнением алгоритма будет передано на распознаватель вхождения $РВ_6$.

Шестой переход НОРКА → ПОРКА должен обеспечить срабатывание распознавателя вхождения буквы Н ($РВ_6$) и выполнение обычной подстановки Н → П ($ОП_6$). Дальнейшее последовательное распознание вхождений подслов М, С, АШ, Ш, КО и Н в слово ПОРКА не даст положительного результата и управление выполнением алгоритма будет передано на распознаватель $РВ_7$.

Последний (седьмой) переход ПОРКА → ПОЛКА должен обеспечить срабатывание распознавателя $РВ_7$ на вхождение буквы Р и выполнение заключительной подстановки Р → Л ($ОП_7$).

Сконструированный нормальный алгоритм Маркова представим соответствующей граф-схемой:



и упорядоченным множеством подстановок:

1) М → К;
2) С → Ш;
3) АШ → ОШ;
4) Ш → Р;
5) КО → НО;
6) Н → П;
7) Р → • Л.

Опишем всю работу нормального алгоритма Маркова от входа «МАСКА» до выхода «ПОЛКА» последовательностью посещенных вершин его граф-схемы с соответствующими комментариями:

Вход «МАСКА»

$РВ_1$ «М», $ОП_1$ «М → К»: «МАСКА» → «КАСКА»

$РВ_1$ «М», $РВ_2$ «С», $ОП_2$ «С → Ш»: «КАСКА» → «КАШКА»

$РВ_1$ «М», $РВ_2$ «С», $РВ_3$ «АШ», $ОП_3$ «АШ → ОШ»: «КАШКА» → → «КОШКА»

$РВ_1$ «М», $РВ_2$ «С», $РВ_3$ «АШ», $РВ_4$ «Ш», $ОП_4$ «Ш → Р»: «КОШКА» → → «КОРКА»

$РВ_1$ «М», $РВ_2$ «С», $РВ_3$ «АШ», $РВ_4$ «Ш», $РВ_5$ «КО», $ОП_5$ «КО → НО»: «КОРКА» → «НОРКА»

$РВ_1$ «М», $РВ_2$ «С», $РВ_3$ «АШ», $РВ_4$ «Ш», $РВ_5$ «КО», $РВ_6$ «Н», $ОП_6$ «Н → → П»: «НОРКА» → «ПОРКА»

$РВ_1$ «М», $РВ_2$ «С», $РВ_3$ «АШ», $РВ_4$ «Ш», $РВ_5$ «КО», $РВ_6$ «Н», $РВ_7$ «Р», $ОП_7$ «Р → Л»: «ПОРКА» → «ПОЛКА»

Выход «ПОЛКА»

Лабораторная работа № 8

РАЗРАБОТКА И ПРОГРАММИРОВАНИЕ АЛГОРИТМА ЛИНЕЙНОЙ СТРУКТУРЫ

Цель: знакомство с простейшим алгоритмом и программой для вычисления значения переменной по формуле.

Задание. Разработать алгоритм вычисления значения переменной y по заданной в табл. 8.1 формуле для вводимых значений переменных a, b и c. Алгоритм представить в виде блок-схемы и программы для ЭВМ на алгоритмическом языке Си.

Основные положения

Алгоритмом называется совокупность правил, обладающих свойствами массовости (инвариантности относительно входной информации), детерминированности (однозначности применения этих правил на каждом шаге), результативности (получения после применения этих правил информации, являющейся результатом) и элементарности (отсутствии необходимости дальнейшего уточнения правил) [3].

Таблица 8.1

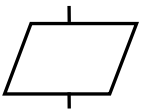
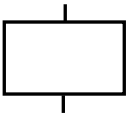
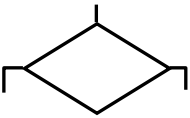
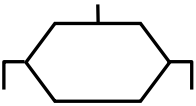
Вариант	Формула	Вариант	Формула
1	$y = \dfrac{2a - 3\ln(b^2 + 1)}{b(1 + \cos^2 c)}$	2	$y = \dfrac{\operatorname{tg}(a^3 - \lvert b\rvert)}{c\,e^b}$
3	$y = \dfrac{2\cos^2 c^3 + 3a}{a(\lvert b\cos a\rvert + 1)}$	4	$y = \dfrac{a\lvert b + c\rvert + \ln b^2}{a\lg^2(c+1)}$
5	$y = \dfrac{\sqrt{\lvert a\rvert^b - 2}}{b(c^3 - 1)}$	6	$y = \dfrac{a - 2\sin^3 b}{a(\operatorname{tg} c + b)}$
7	$y = \dfrac{e^a + \lvert b\rvert}{a\ln(c^4 + 1)}$	8	$y = \dfrac{\sqrt{2^{a+b}} + \arccos c}{b\operatorname{arctg}\dfrac{b}{c}}$
9	$y = \dfrac{\lg(\lvert a\rvert + 3)^c}{b \cdot 10^a}$	10	$y = \dfrac{\sqrt{b + \cos^2 a} + e^c}{\lvert a + b + c\rvert}$
11	$y = \dfrac{\sin^3(a+b)^2}{a^2(b+c)}$	12	$y = \dfrac{\ln(a^2 + b^2) + \lg c}{b(a^b + c)}$
13	$y = \dfrac{\sqrt{e^a} + \operatorname{tg} b^2}{(a+b)c}$	14	$y = \dfrac{\sin 10^a + \sqrt{b^2 + c^2}}{a\operatorname{tg}\dfrac{b}{c}}$
15	$y = \dfrac{\arccos^2 a - b^2}{b(c-1)}$	16	$y = \dfrac{\arccos^2 b - e^{a+b}}{\lvert a\rvert(b+c)}$
17	$y = \dfrac{\operatorname{arctg}(a^2 + 1) + \cos b}{a\,e^c}$	18	$y = \dfrac{\operatorname{arctg}(b^2 + 1) + \cos^3 a}{(a+b)e^{\lvert b+c\rvert}}$
19	$y = \dfrac{\ln\lvert a\rvert + b^c}{c(10^a - b^2)}$	20	$y = \dfrac{\ln a^b + 10^{b+c}}{a\sqrt{c^2 + 1}}$
21	$y = \dfrac{\sqrt{\sin^2 a^3}}{a\operatorname{arctg}\dfrac{b}{c}}$	22	$y = \dfrac{\operatorname{tg}^2 a - b\,c}{b\arccos(b^2 + c^2)}$
25	$y = \dfrac{\sqrt{\arcsin^2 b + 1}}{c(e^a + \lvert b\rvert)}$	26	$y = \dfrac{e^c + \lvert b^2 - a\rvert}{a\ln^2(b+c)}$
27	$y = \dfrac{\ln(a^2 + b^2) + b^c}{a\,b\,c}$	28	$y = \dfrac{\lg a^3 + b^{a+c}}{(a+b)10^c}$

Продолжение табл. 8.1

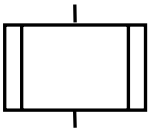
Вариант	Формула	Вариант	Формула
29	$y=\dfrac{10^{a+b}+\sin^3 c}{b\sqrt{a^2+b^2}}$	30	$y=\dfrac{\sin\sqrt{a+b}}{a\,\mathrm{tg}^2(b+c)}$

Одним из способов представления алгоритмов является изображение их с помощью так называемых блок-схем [5]. Под *блок-схемой* алгоритма понимается графическое представление алгоритма с помощью специальных блоков, соединённых между собой линиями передачи управления. Наиболее часто используемые блоки приведены в табл. 8.2.

Таблица 8.2

Название блока	Обозначение	Пояснение
Пуск-останов		Начало, конец, прерывание процесса обработки данных или выполнения программы
Ввод-вывод		Преобразование данных в форму, пригодную для обработки (ввод) или отображения результатов обработки (вывод)
Процесс		Выполнение операции или группы операций, в результате которых изменяется значение, форма представления или расположение данных
Решение		Выбор направления выполнения алгоритма или программы в зависимости от некоторых переменных условий
Модификация		Выполнение операций, меняющих команды или группы команд, изменяющих программу

Продолжение табл. 8.2

Название блока	Обозначение	Пояснение
Предопределённый процесс		Использование ранее созданных и отдельно описанных алгоритмов или программ

Краткость и выразительность блок-схем позволяет разрабатывать алгоритмы высокого качества. В алгоритме вычисления значения переменной по заданной формуле необходимо предусмотреть ввод исходных данных, вычисление значения переменной и вывод полученного результата.

Заметим, что представленный блок-схемой алгоритм не может непосредственно восприниматься вычислительной машиной. Ввести алгоритм в память ЭВМ, которая затем выполнит заданные им предписания и получит искомые результаты, можно в виде *программы*, написанной на одном из алгоритмических языков.

В программе заданная формула выступает в виде *оператора присваивания*, который устанавливает значение переменной, равное предварительно вычисленному значению арифметического выражения. При этом арифметическое выражение представляется *линейной записью*, под которой понимается запись выражения в одну строку с применением соответствующих выбранному алгоритмическому языку знаков операций сложение (+), вычитание (–), умножение (*), деление (/) и *стандартных математических функций*.

Некоторые стандартные математические функции языка Си приведены в табл. 8.3.

Таблица 8.3

Имя	Прототип	Вычисляемое значение
acos	*double acos* (*double x*)	arccos *x* (в радианах)
asin	*double asin* (*double x*)	arcsin *x* (в радианах)
atan	*double atan* (*double x*)	arctg *x* (в радианах)
*atan*2	*double atan*2 (*double x*, *double y*)	arctg *x*/*y* (в радианах)
cos	*double cos* (*double x*)	cos *x* (*x* в радианах)
exp	*double exp* (*double x*)	e^x
fabs	*double fabs* (*double x*)	$\lvert x \rvert$

Продолжение табл. 8.3

Имя	Прототип	Вычисляемое значение
log	*double log* (*double x*)	$\ln x$ ($x > 0$)
*log*10	*double log*10 (*double x*)	$\lg x$ ($x > 0$)
pow	*double pow* (*double x*, *double y*)	x^y ($x \geq 0$)
sin	*double sin* (*double x*)	$\sin x$ (x в радианах)
sqrt	*double sqrt* (*double x*)	$\sqrt{x}$ ($x \geq 0$)
tan	*double tan* (*double x*)	$\operatorname{tg} x$ (x в радианах)

Описание этих функций находится в файле *math.h*, который следует подключать к тексту программы.

Для составления линейной записи арифметического выражения необходимо сначала построить бинарное дерево, прохождение которого в концевом порядке (левое поддерево, правое поддерево, корень) [6] соответствует последовательности вычисления значения этого выражения. Затем, обходя бинарное дерево в обратном порядке (левое поддерево, корень, правое поддерево), получить искомую линейную запись. При этом следует помнить, что в вызове какой-либо стандартной функции в первую очередь указывается её имя, а во вторую – список аргументов в круглых скобках. Наконец, если порядок выполнения действий, определяемый приоритетом операций (вызов функции, умножение и деление, сложение и вычитание), не соответствует правильной последовательности вычисления значения арифметического выражения, в его линейную запись необходимо добавить круглые скобки.

Пример выполнения задания

Задание. Разработать алгоритм вычисления значения переменной y по заданной формуле

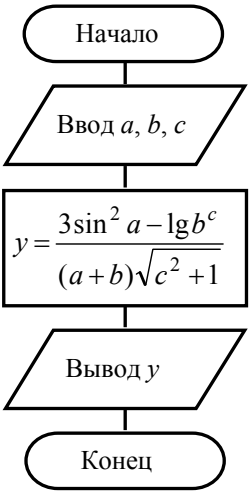
$$y = \frac{3\sin^2 a - \lg b^c}{(a+b)\sqrt{c^2+1}}$$

для вводимых значений переменных a, b и c. Алгоритм представить в виде блок-схемы и программы для ЭВМ на алгоритмическом языке Си.

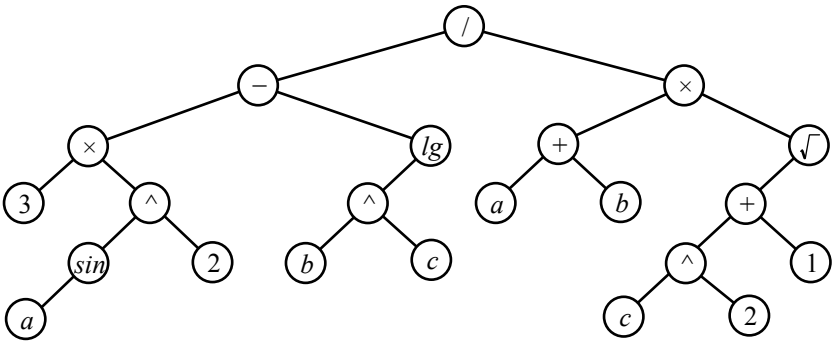
Решение.

В алгоритме вычисления значения переменной y по формуле предусмотрим: ввод исходных данных (значений переменных a, b и c), вычисление значения переменной y и вывод полученного результата.

Блок-схема алгоритма будет иметь следующий вид:



Для составления линейной записи арифметического выражения $\frac{3\sin^2 a - \lg b^c}{(a+b)\sqrt{c^2+1}}$ построим бинарное дерево, прохождение которого в концевом порядке (левое поддерево, правое поддерево, корень) соответствует последовательности вычисления значения этого выражения:



Обходя это бинарное дерево в обратном порядке (левое поддерево, корень, правое поддерево), получить искомую линейную запись применительно к языку Си:

3 * *pow* (*sin* (*a*), 2) – *log*10 (*pow* (*b*, *c*)) / *a* + *b* * *sqrt* (*pow* (*c*, 2) + 1).

Поскольку порядок выполнения действий, определяемый приоритетом операций (вызов функции, умножение и деление, сложение и вычитание), не соответствует правильной последовательности вычисления значения арифметического выражения, в линейную запись добавим необходимые круглые скобки:

(3 * *pow* (*sin* (*a*), 2) – *log*10 (*pow* (*b*, *c*))) / ((*a* + *b*) * *sqrt* (*pow* (*c*, 2) + 1)).

Полученную линейную запись включим в оператор присваивания программы на алгоритмическом языке Си, которая будет иметь следующий вид:

```
#include<stdio.h>
#include<math.h>
void main()
{
   double a,b,c,y;
   scanf("%le%le%le",&a,&b,&c);
   y=(3*pow(sin(a),2)-
      log10(pow(b,c)))/((a+b)*sqrt(pow(c,2)+1));
   printf("%le\n",y);
}
```

Приведём тестовый пример для данной программы:

Ввод			Вывод
a	*b*	*c*	*y*
1	2	3	1,287184e-01

Заметим, что полученное значение 1,287184e-01 следует понимать как $1{,}287184 \cdot 10^{-1}$.

Лабораторная работа № 9

РАЗРАБОТКА И ПРОГРАММИРОВАНИЕ АЛГОРИТМА РАЗВЕТВЛЁННОЙ СТРУКТУРЫ

Цель работы: приобретение навыков организации ветвлений в алгоритмах.

Задание. Разработать алгоритм определения количества чисел, обладающих некоторым общим свойством, среди заданных целых чисел *a* и *b* или *a*, *b* и *c* (табл. 9.1). Представить этот алгоритм в виде блок-схемы и программы для ЭВМ на алгоритмическом языке Си.

Таблица 9.1

Вариант	Определить количество	среди чисел
1	положительных чисел	*a*, *b*, *c*
2	отрицательных чисел	*a*, *b*
3	отрицательных чисел	*a*, *b*, *c*
4	неположительных чисел	*a*, *b*
5	неположительных чисел	*a*, *b*, *c*
6	неотрицательных чисел	*a*, *b*
7	неотрицательных чисел	*a*, *b*, *c*
8	равных нулю чисел	*a*, *b*
9	равных нулю чисел	*a*, *b*, *c*
10	не равных нулю чисел	*a*, *b*
11	не равных нулю чисел	*a*, *b*, *c*
12	чётных чисел	*a*, *b*
13	чётных чисел	*a*, *b*, *c*
14	нечётных чисел	*a*, *b*
15	нечётных чисел	*a*, *b*, *c*
16	чисел, кратных трём	*a*, *b*
17	чисел, кратных трём	*a*, *b*, *c*
18	чисел, не кратных трём	*a*, *b*
19	чисел, не кратных трём	*a*, *b*, *c*
20	чисел, больших числа *c*	*a*, *b*
21	чисел, больших числа *d*	*a*, *b*, *c*
22	чисел, меньших числа *c*	*a*, *b*
23	чисел, меньших числа *d*	*a*, *b*, *c*
24	чисел, не больших числа *c*	*a*, *b*
25	чисел, не больших числа *d*	*a*, *b*, *c*
26	чисел, не меньших числа *c*	*a*, *b*
27	чисел, не меньших числа *d*	*a*, *b*, *c*
28	чисел, равных числу *c*	*a*, *b*
29	чисел, равных числу *d*	*a*, *b*, *c*
30	чисел, не равных числу *c*	*a*, *b*

Основные положения

Под *ветвлением* в алгоритме понимается организация выбора одного из двух альтернативных вариантов продолжения алгоритма в соответствии со значением некоторого логического выражения. Каждый из упомянутых альтернативных вариантов называется *ветвью* в алгоритме.

Организовать ветвления в алгоритме решения поставленной задачи можно по-разному. В одном случае: определить количество n вариантов возможных значений искомой величины; составить для каждого из этих n вариантов соответствующее логическое выражение; организовать в алгоритме n ветвей с помощью $(n - 1)$ ветвления, используя при этом $(n - 1)$ логическое выражение; в каждой ветви искомой величине присвоить необходимое значение. Другой случай требует задания нулевого начального значения искомой величины и организации $(n - 1)$ ветвлений алгоритма таким образом, чтобы эта искомая величина претерпевала поэтапные изменения до своего конечного значения.

В блок-схемах алгоритмов ветвления реализуются с помощью блока «Решение» (см. табл. 8.2). При этом внутри блока записывается логическое выражение, а выходящие ветви помечаются словами «Да» и «Нет», соответствующими истинному и ложному значениям логического выражения.

В программах на алгоритмическом языке Си ветвления организуются с помощью условных операторов `if` или `if else`. Синтаксис оператора `if` имеет вид [7]:

```
if (выражение)
оператор
```

Оператору `if` соответствует базовая управляющая конструкция, представленная на рис. 9.1.

Из неё видно, что если выражение в заголовке условного оператора вырабатывает истинное значение, то `оператор` выполняется, в противном случае управление передаётся оператору, следующему за условным.

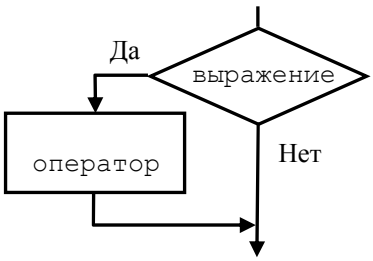


Рис. 9.1

Синтаксис оператора `if else` имеет следующий вид:

```
if (выражение)
оператор1
else
оператор2
```

Такой условный оператор реализует базовую управляющую конструкцию, изображённую на рис. 9.2.

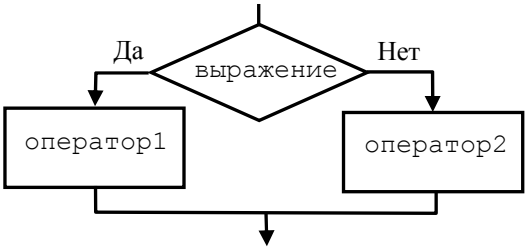


Рис. 9.2

Из этой конструкции следует, что если выражение истинно, то выполняется `оператор1`, а в противном случае – `оператор2`. Затем управление передаётся оператору, следующему за условным оператором.

При представлении логических выражений в разрабатываемых программах необходимо учитывать следующие соответствия обозначений операций отношения и логических операций на языке математики и алгоритмическом языке Си (табл. 9.2).

Таблица 9.2

Язык	Операции отношения						Логические операции		
Математика	<	>	≤	≥	=	≠	**не**	**или**	**и**
Си	<	>	<=	>=	==	!=	!	||	&&

Установление чётности и нечётности целого числа в программах на языке Си можно осуществить с помощью операции определения остатка от целочисленного деления, обозначаемой символом %. Так, например, целое число *a* является чётным, если значение *a*%2 (остаток от деления *a* на 2) равно нулю, и нечётным – в противном случае.

Пример выполнения задания

Задание. Разработать алгоритм определения количества положительных чисел среди заданных целых чисел *a*, *b*; представить алгоритм в виде блок-схемы и программы для ЭВМ на алгоритмическом языке Си.

Решение 1.

Искомую величину – количество положительных чисел среди заданных чисел *a*, *b* – обозначим именем *k*. Переменная *k* в нашей задаче может принимать только значения 0, 1 и 2, поэтому $n = 3$.

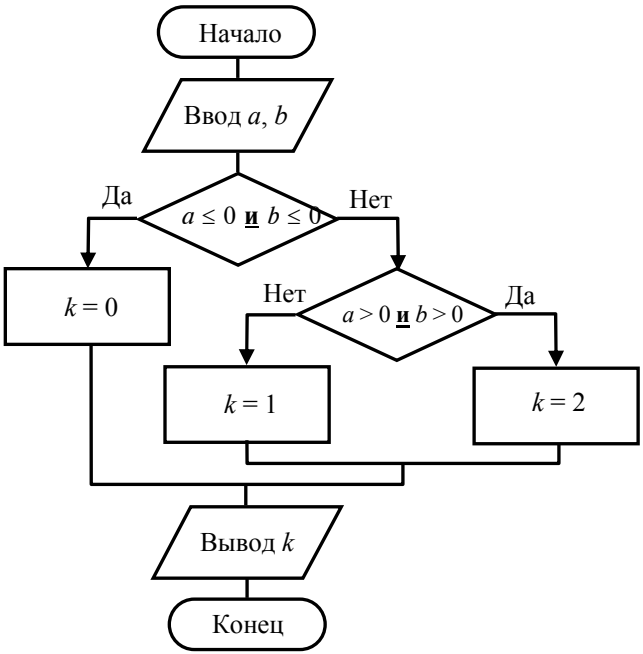
Составим для каждого из трёх возможных вариантов значения искомой величины *k* логическое выражение, истинное только для него и только для него:

$k = 0 \leftrightarrow a \le 0$ **и** $b \le 0$,

$k = 1 \leftrightarrow a > 0$ **и** $b \le 0$ **или** $a \le 0$ **и** $b > 0$,

$k = 2 \leftrightarrow a > 0$ **и** $b > 0$.

Организуем в алгоритме три ветви с помощью двух ветвлений, используя два логических выражения, и в каждой ветви присвоим искомой величине необходимое значение. Блок-схема алгоритма будет иметь следующий вид:



Программа на алгоритмическом языке Си будет выглядеть следующим образом:

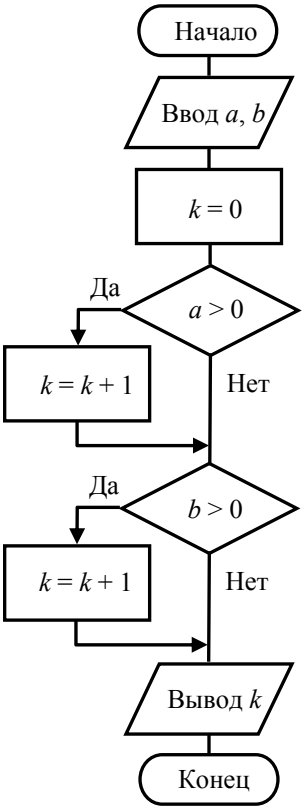
```
#include<stdio.h>
void main()
{
   int a,b,k;
   scanf("%d%d",&a,&b);
   if (a<=0 && b<=0)
```

```
    k=0;
    else
    if (a>0 && b>0)
    k=2;
    else
    k=1;
    printf("%d\n",k);
}
```

Решение 2.

Искомую величину – количество положительных чисел среди заданных чисел *a*, *b* – обозначим именем *k*. Переменная *k* в нашей задаче может принимать только значения 0, 1 и 2, поэтому $n = 3$.

Зададим в качестве начального значения искомой величины *k* число ноль, и два ветвления в алгоритме организуем таким образом, чтобы *k* претерпевала поэтапные изменения до своего конечного значения. Блок-схема алгоритма будет иметь следующий вид:



Программа на алгоритмическом языке Си будет выглядеть следующим образом:

```
#include<stdio.h>
void main()
{
   int a,b,k;
   scanf("%d%d",&a,&b);
   k=0;
   if (a>0)
   k=k+1;
   if (b>0)
   k=k+1;
   printf("%d\n",k);
}
```

Приведём тестовые примеры для разработанных в решениях 1 и 2 программ:

Ввод		Вывод
a	*b*	*k*
0	−1	0
2	0	1
0	3	1
3	5	2

Лабораторная работа № 10

РАЗРАБОТКА И ПРОГРАММИРОВАНИЕ ЦИКЛИЧЕСКОГО АЛГОРИТМА

Цель: приобретение навыков программирования алгоритма, содержащего цикл с параметром.

Задание. По заданной в табл. 10.1 числовой последовательности получить соответствующую рекуррентную формулу; разработать алгоритм вычисления суммы *n* первых элементов этой последовательности; представить алгоритм в виде блок-схемы и программы для ЭВМ на алгоритмическом языке Си.

Таблица 10.1

Вариант	Последовательность	Вариант	Последовательность
1	1, 7, 37, 187, 937, ...	2	3, 14, 58, 234, 938, ...
3	4, 18, 74, 298, 1194, ...	4	2, 13, 68, 343, 1718, ...
5	2, 10, 42, 170, 682, ...	6	1, 5, 17, 53, 161, ...
7	3, 12, 39, 120, 363, ...	8	4, 14, 44, 134, 404, ...
9	1, 6, 16, 36, 76, ...	10	4, 12, 28, 60, 124, ...
11	4, 21, 106, 531, 2656, ...	12	1, 7, 25, 79, 241, ...
13	4, 17, 69, 277, 1109, ...	14	2, 9, 37, 149, 597, ...
15	3, 15, 63, 255, 1023, ...	16	4, 20, 84, 340, 1364, ...
17	2, 14, 74, 374, 1874, ...	18	3, 7, 15, 31, 63, ...
19	3, 8, 18, 38, 78, ...	20	4, 24, 124, 624, 3124, ...
21	2, 12, 62, 312, 1562, ...	22	4, 23, 118, 593, 2968, ...
23	4, 19, 79, 319, 1279, ...	24	1, 6, 31, 156, 781, ...
25	1, 4, 13, 40, 121, ...	26	3, 10, 24, 52, 108, ...
27	2, 12, 52, 212, 852, ...	28	2, 8, 26, 80, 242, ...
29	3, 19, 99, 499, 2499, ...	30	4, 10, 22, 46, 94, ...

Основные положения

Рекуррентной формулой называется формула, которая связывает между собой (p + 1) соседних элементов некоторой числовой последовательности. Зная p первых элементов этой числовой последовательности, можно с помощью такой формулы шаг за шагом последовательно вычислить (p + 1)-й, (p + 2)-й, (p + 3)-й, … элементы. Заметим, что все заданные в табл. 10.1 последовательности вида a_1, a_2, a_3, a_4, a_5, … получены с применением рекуррентной формулы $a_i = ba_{i-1} + c$, где b и c – некоторые коэффициенты. Эта формула связывает два соседних элемента последовательности a_i и a_{i-1} и, следовательно, $p + 1 = 2$, а $p = 1$. Таким образом, зная только первый элемент последовательности, можно вычислить второй, третий, четвёртый и т.д. Для определения значений коэффициентов b и c в рекуррентной формуле достаточно решить систему двух линейных уравнений $\begin{cases} a_2 = ba_1 + c; \\ a_3 = ba_2 + c. \end{cases}$

Алгоритм вычисления суммы s первых n элементов числовой последовательности можно организовать следующим образом. Ввести значение n; задать значение a_1 и начальное значение s, равное a_1; для каждого значения i от 2 до n вычислить a_i по рекуррентной формуле $a_i = ba_{i-1} + c$ и увеличить сумму s на значение этого элемента последовательности; вывести полученное в итоге значение суммы s.

Поскольку в данной задаче после прибавления очередного элемента последовательности к сумме он необходим только для вычисления следующего элемента, индексы элементов последовательности можно опустить и тогда рекуррентная формула примет вид $a = ba + c$.

Повторяющиеся действия (циклы) в алгоритме, как правило, организуются с помощью некоторого изменяющегося параметра, называемого *параметром цикла*. В блок-схемах алгоритмов цикл с начальным значением `in`, заранее известным конечным значением `ik` и шагом изменения `ih` параметра `i` может быть представлен базовой управляющей конструкцией, представленной на рис. 10.1.

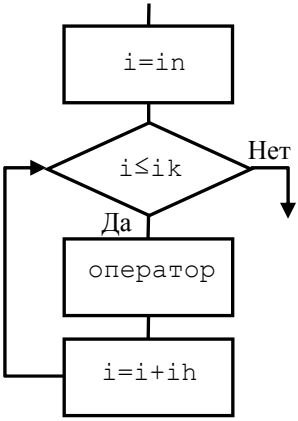


Рис. 10.1

Эту конструкцию часто в блок-схемах изображают с помощью блока «Подготовка» в более компактном виде, как на рис. 10.2.

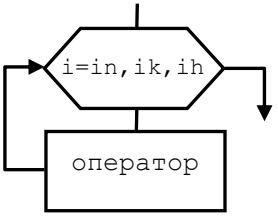


Рис. 10.2

В программах на алгоритмическом языке Си подобные циклы с параметром реализуются с помощью оператора `for`, имеющего следующий синтаксис:

```
for(i=in;i<=ik;i+=ih)
оператор
```

Пример выполнения задания

По заданной числовой последовательности 5, 11, 23, 47, 95, … получить соответствующую рекуррентную формулу; разработать алгоритм вычисления суммы n первых элементов этой последовательности; представить алгоритм в виде блок-схемы и программы для ЭВМ на алгоритмическом языке Си.

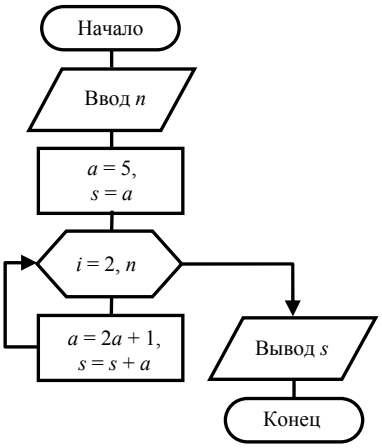
Решение.

Известно, что заданная последовательность 5, 11, 23, 47, 95, … получена с применением рекуррентной формулы $a_i = ba_{i-1} + c$. Поэтому для определения значений коэффициентов b и c этой формулы решим систему уравнений вида $\begin{cases} a_2 = ba_1 + c; \\ a_3 = ba_2 + c: \end{cases}$

$$\begin{cases} 11 = 5b + c; \\ 23 = 11b + c; \end{cases} \quad \begin{cases} 11 = 5b + c; \\ 12 = 6b; \end{cases} \quad \begin{cases} 11 = 5b + c; \\ b = 2; \end{cases} \quad \begin{cases} 11 = 10 + c; \\ b = 2; \end{cases} \quad \begin{cases} c = 1; \\ b = 2. \end{cases}$$

Таким образом, получена рекуррентная формула $a_i = 2a_{i-1} + 1$, которая после опускания индексов элементов примет вид $a = 2a + 1$.

Алгоритм вычисления суммы s первых n элементов числовой последовательности организуем следующим образом. Введём значение n; зададим значение $a = 5$ и начальное значение s, равное a; для каждого значения i от 2 до n вычислим следующее значение a по формуле $a = 2a + 1$ и увеличим сумму s на значение a; выведем полученное в итоге значение суммы s. Блок-схема алгоритма будет иметь следующий вид:



Программа на алгоритмическом языке Си:

```
#include<stdio.h>
void main()
{
   int n,a,s,i;
   scanf("%d",&n);
   a=5;
   s=a;
   for(i=2;i<=n;i++)
   {
      a=2*a+1;
      s=s+a;
   }
   printf("%d\n",s);
}
```

Приведём тестовые примеры для разработанной программы:

Ввод	*n*	1	2	3	4
Вывод	*s*	5	16	39	86

Лабораторная работа № 11

АЛГОРИТМ И ПРОГРАММА ОПРЕДЕЛЕНИЯ СУММЫ ЧИСЛОВОГО РЯДА

Цель: приобретение навыков алгоритмизации вычисления суммы числового ряда с заданной точностью.

Задание. По заданному в табл. 11.1 числовому ряду разработать алгоритм вычисления его суммы с точностью ε; представить алгоритм в виде блок-схемы и программы для ЭВМ на алгоритмическом языке Си.

Основные положения

Алгоритм вычисления суммы *s* числового ряда с точностью ε можно организовать следующим образом. Ввести значение точности ε. Задать начальное значение суммы *s*, равное 0, и начальное значение параметра цикла *k*, равное 0, 1 или 2 (в зависимости от конкретного числового ряда). Используя общий вид слагаемого, вычислить значение первого слагаемого *sl* для начального значения *k*.

Таблица 11.1

Вариант	Числовой ряд	Вариант	Числовой ряд
1	$\sum_{k=1}^{\infty}(-1)^{k+1}\frac{1}{k}=\ln 2$	2	$\sum_{k=1}^{\infty}(-1)^{k+1}\frac{1}{2k-1}=\frac{\pi}{4}$
3	$\sum_{k=1}^{\infty}(-1)^{k+1}\frac{1}{k^2}=\frac{\pi^2}{12}$	4	$\sum_{k=1}^{\infty}\frac{1}{(2k-1)^2}=\frac{\pi^2}{8}$
5	$\sum_{k=1}^{\infty}\frac{(-1)^{k+1}}{(2k-1)^3}=\frac{\pi^3}{32}$	6	$\sum_{k=1}^{\infty}\frac{1}{(2k-1)^4}=\frac{\pi^4}{96}$
7	$\sum_{k=1}^{\infty}\frac{(-1)^{k+1}}{(2k-1)^5}=\frac{5\pi^5}{1536}$	8	$\sum_{k=1}^{\infty}(-1)^{k+1}\frac{k}{(k+1)^2}=\frac{\pi^2}{12}-\ln 2$
9	$\sum_{k=1}^{\infty}\frac{1}{4k^2-1}=\frac{1}{2}$	10	$\sum_{k=1}^{\infty}\frac{1}{(4k^2-1)^2}=\frac{\pi^2-8}{16}$
11	$\sum_{k=1}^{\infty}\frac{1}{(4k^2-1)^3}=\frac{32-3\pi^2}{64}$	12	$\sum_{k=1}^{\infty}\frac{1}{(4k^2-1)^4}=\frac{\pi^4+30\pi^2-384}{768}$
13	$\sum_{k=1}^{\infty}\frac{1}{k(4k^2-1)}=2\ln 2-1$	14	$\sum_{k=1}^{\infty}\frac{1}{k(9k^2-1)}=\frac{3}{2}(\ln 3-1)$
15	$\sum_{k=1}^{\infty}\frac{k}{(4k^2-1)^2}=\frac{1}{8}$	16	$\sum_{k=1}^{\infty}\frac{1}{k(36k^2-1)}=-3+\frac{3}{2}\ln 3+2\ln 2$
17	$\sum_{k=1}^{\infty}\frac{1}{k(4k^2-1)^2}=\frac{3}{2}-2\ln 2$	18	$\sum_{k=1}^{\infty}\frac{12k^2-1}{k(4k^2-1)^2}=2\ln 2$
19	$\sum_{k=1}^{\infty}\frac{1}{(2k-1)(2k+1)}=\frac{1}{2}$	20	$\sum_{k=1}^{\infty}\frac{1}{(4k-1)(4k+1)}=\frac{1}{2}-\frac{\pi}{8}$
21	$\sum_{k=2}^{\infty}\frac{1}{(k-1)(k+1)}=\frac{3}{4}$	22	$\sum_{k=1}^{\infty}\frac{1}{(2k-1)2k(2k+1)}=\ln 2-\frac{1}{2}$
23	$\sum_{k=1}^{\infty}\frac{(-1)^{k+1}}{(2k-1)2k(2k+1)}=$ $=\frac{1}{2}(1-\ln 2)$	24	$\sum_{k=0}^{\infty}\frac{1}{(3k+1)(3k+2)(3k+3)(3k+4)}=$ $=\frac{1}{6}-\frac{1}{4}\ln 3+\frac{\pi}{12\sqrt{3}}$

Продолжение табл. 11.1

Вариант	Числовой ряд	Вариант	Числовой ряд
25	$\sum_{k=1}^{\infty}(-1)^{k+1}\frac{1}{3k-2}=\frac{1}{3}\left(\frac{\pi}{\sqrt{3}}+\ln 2\right)$	26	$\sum_{k=1}^{\infty}(-1)^{k+1}\frac{1}{3k-1}=\frac{1}{3}\left(\frac{\pi}{\sqrt{3}}-\ln 2\right)$
27	$\sum_{k=1}^{\infty}\frac{1}{(8k-1)(8k+1)}=$ $=\frac{1}{2}-\frac{\pi}{16}(\sqrt{2}+1)$	28	$\sum_{k=1}^{\infty}(-1)^{k+1}\frac{1}{4k-3}=$ $=\frac{1}{4\sqrt{2}}(\pi+2\ln(\sqrt{2}+1))$
29	$\sum_{k=1}^{\infty}\frac{1}{2^k k}=\ln 2$	30	$\sum_{k=1}^{\infty}\frac{1}{2^k k^2}=\frac{\pi^2}{12}-\frac{1}{2}(\ln 2)^2$

Далее в цикле, пока величина слагаемого (абсолютная величина слагаемого для знакочередующихся рядов) больше ε, повторять следующую последовательность действий: увеличить значение суммы *s* на значение слагаемого *sl*, увеличить значение *k* на 1 и определить следующее значение слагаемого *sl*. После окончания цикла вычислить предельное значение суммы ряда *sr*, используя при этом выражение, которое записано после знака равенства в заданном числовом ряде. И, наконец, вывести полученные значения суммы *s* и предельное значение суммы ряда *sr*.

В циклах типа «пока» проверка условия производится перед выполнением тела цикла. Если результат вычисления условного выражения не равен нулю (значение «истина»), то выполняется тело цикла и осуществляется возврат на проверку значения условного выражения; в противном случае цикл заканчивается. Перед входом в цикл «пока» обычно инициализируют одну или несколько переменных для того, чтобы условное выражение имело какое-либо значение. В теле цикла, как правило, должны изменяться значения одной или нескольких переменных, входящих в условное выражение, чтобы в конце концов выражение обратилось в ноль (значение «ложь») и цикл завершился.

В блок-схемах алгоритмов цикл «пока» представляется базовой управляющей конструкцией, представленной на рис. 11.1.

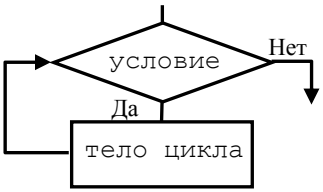


Рис. 11.1

В программах на алгоритмическом языке Си такие циклы реализуются с помощью оператора `while`, имеющего следующий синтаксис [7]:

```
while(условие)
тело цикла
```

Заметим, что в языке Си деление целого числа на целое даёт целое число. Поэтому параметр цикла k в программах целесообразно описать как вещественную переменную. Из тех же соображений в дробях с целыми числителем и знаменателем следует использовать не целые, а вещественные константы. Для представления числа π следует использовать константу `M_PI`, которая описана в файле `math.h`, а буквы греческого алфавита заменять словами, составленными из латинских букв.

Пример выполнения задания

Задание. По заданному числовому ряду $\sum_{k=0}^{\infty}(-1)^k \frac{1}{3^{2k}} = \frac{9}{10}$ разработать алгоритм вычисления его суммы с точностью ε; представить алгоритм в виде блок-схемы и программы для ЭВМ на алгоритмическом языке С.

Решение.

Алгоритм вычисления суммы s числового ряда с точностью ε представим следующей схемой:

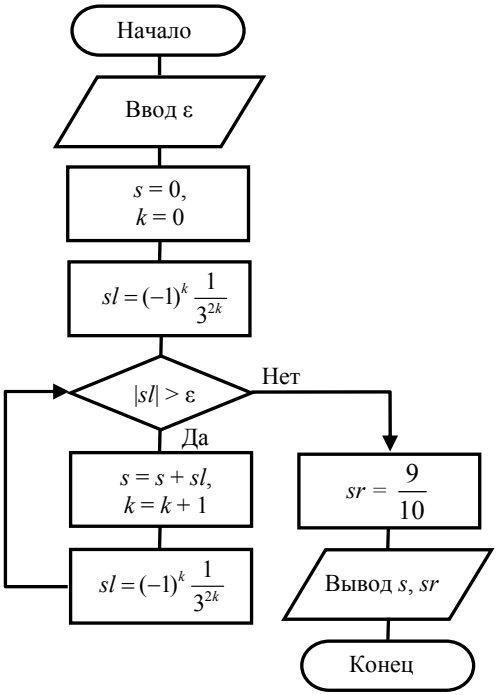


Схема предполагает ввод значения точности ε. Задание начального значения суммы *s*, равного 0, и начального значения параметра цикла *k*, равного 0. Вычисление значения слагаемого *sl* для начального значения *k* (первого слагаемого) по формуле $sl = (-1)^k \frac{1}{3^{2k}}$. Далее в цикле, пока абсолютная величина слагаемого *sl* (наш ряд знакочередующийся) больше ε: увеличение значение суммы *s* на значение слагаемого *sl*, увеличение значения *k* на 1 и определение следующего значения слагаемого *sl* по той же формуле. После окончания цикла вычисляется предельное значение суммы ряда *sr* и выводятся полученные значения *s* и *sr*.

В программе на языке Си для представления переменной ε будем использовать идентификатор `eps`, все переменные опишем как вещественные восьмибайтные (тип `double`), а дробь $\frac{9}{10}$ представим выражением с использованием вещественных констант `9,0 и 10,0`:

```
#include<stdio.h>
#include<math.h>
void main()
{
   double eps,s,k,sl,sr;
   scanf("%le",&eps);
   s=0; k=0;
   sl=pow(-1,k)/pow(3,2*k);
   while(fabs(sl)>eps)
   {
      s+=sl; k++;
      sl=pow(-1,k)/pow(3,2*k);
   }
   sr=9.0/10.0;
   printf("s=%lf sr=%lf\n",s,sr);
}
```

Результаты вычислений по программе внесём в таблицу:

Ввод	eps	1e–1	1e–2	1e–3	1e–6
Вывод	s	0,888889	0,901235	0,899863	0,900000
	sr	0,900000	0,900000	0,900000	0,900000

Лабораторная работа № 12

РАЗРАБОТКА И ПРОГРАММИРОВАНИЕ АЛГОРИТМА ОБРАБОТКИ ОДНОМЕРНОГО МАССИВА

Цель: знакомство с понятием одномерного массива и приобретение навыков организации его обработки.

Задание. Разработать алгоритм обработки одномерного массива из n элементов, заполненного целыми случайными числами из диапазона от a до b, в соответствии с поставленной задачей (табл. 12.1). Представить алгоритм в виде блок-схемы и программы для ЭВМ на алгоритмическом языке Си.

Таблица 12.1

Вариант	Задача обработки массива заключается в определении
1	суммы всех положительных элементов
2	суммы всех отрицательных элементов
3	суммы всех чётных элементов
4	суммы всех нечётных элементов
5	суммы всех кратных трём элементов
6	суммы всех не кратных трём элементов
7	произведения всех положительных элементов
8	произведения всех отрицательных элементов
9	произведения всех чётных элементов
10	произведения всех нечётных элементов
11	произведения всех кратных трём элементов
12	произведения всех не кратных трём элементов
13	количества всех положительных элементов
14	количества всех отрицательных элементов
15	количества всех чётных элементов
16	количества всех нечётных элементов
17	количества всех кратных трём элементов
18	количества всех не кратных трём элементов
19	минимального элемента
20	минимального элемента из всех положительных элементов

Продолжение табл. 12.1

Вариант	Задача обработки массива заключается в определении
21	минимального элемента из всех чётных элементов
22	минимального элемента из всех нечётных элементов
23	минимального элемента из всех кратных трём элементов
24	минимального элемента из всех не кратных трём элементов
25	максимального элемента
26	максимального элемента из всех отрицательных элементов
27	максимального элемента из всех чётных элементов
28	максимального элемента из всех нечётных элементов
29	максимального элемента из всех кратных трём элементов
30	максимального элемента из всех не кратных трём элементов

Основные положения

Одномерным массивом называется поименованная конечная последовательность объектов одинакового типа (элементов массива). Графически одномерный массив из *n* элементов представляется линейной таблицей с *n* ячейками.

Разрабатываемый алгоритм должен содержать: ввод значений *n*, *a* и *b*; присвоение каждому элементу массива целого случайного числа из диапазона от *a* до *b*; дальнейшие действия, связанные с обработкой массива и выводом результата.

Массивы в языке Си описываются следующим образом:

```
тип имя_массива[размер_массива];
```

При этом компилятор отводит под массив память в количестве `(sizeof(тип) * размер_массива)` байтов, где `sizeof` является операцией, определяющей число байтов, занимаемой операндом `тип`.

Доступ к какому-либо элементу массива (ячейке таблицы) осуществляется посредством указания имени массива (имени таблицы), а также индекса этого элемента (номера ячейки).

Все массивы индексируются, начиная с нуля. Например, если целочисленный массив data описан как

```
int data[10];
```

то под него будет выделено `(sizeof(int) * размер_массива)` = = (2 * 10) = 20 байтов, а десятью элементами типа int будут элементы:

```
data[0], data[1], data[2], …, data[9].
```

Для описания целочисленного массива `data`, состоящего, например, из 10 элементов, можно использовать инструкцию

```
int data[n];
```

с предварительным заданием значения *n* с помощью макроопределения

```
#define n 10
```

которое увязывает идентификатор *n* со значением 10 при любом появлении этого идентификатора в тексте программы.

Под обработкой массива понимается анализ, замена, перестановка его элементов и другие возможные действия.

Для задания случайных чисел в языке Си может быть использован генератор случайных чисел с прототипом `int random(int num)`, который возвращает целое случайное число от `0` до `num-1`, и функция с прототипом `void randomize(void)`, которая инициализирует генератор. Обе эти функции описаны в файле `stdlib.h`.

Установление чётности и нечётности некоторого целого числа в программах на языке Си можно осуществить с помощью операции % определения остатка от целочисленного деления. Так, элемент `data[i]` массива является чётным, если `data[i]%2` (остаток от деления его на 2) равен нулю, и нечётным – в противном случае.

Для определения количества переходов от одного характерного элемента массива к другому в условное выражение следует включить соседние элементы массива, например, `data[i]` и `data[i+1]`.

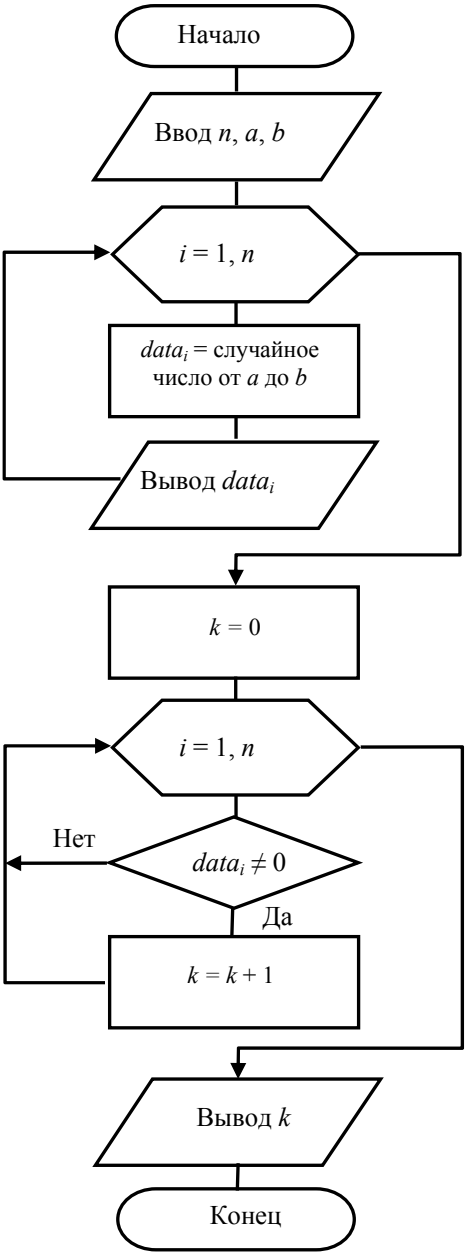
Пример выполнения задания

Задание. Разработать алгоритм обработки одномерного массива из *n* элементов, заполненного целыми случайными числами из диапазона от *a* до *b*, в соответствии с поставленной задачей определения количества ненулевых элементов; представить алгоритм в виде блок-схемы и программы для ЭВМ на алгоритмическом языке Си.

Решение.

В разрабатываемом алгоритме предусмотрим: ввод значений *n*, *a* и *b*; присвоение каждому элементу массива (назовём его `data`) целого случайного числа из диапазона от *a* до *b*; вывод полученного массива `data`; присвоение количеству *k* ненулевых элементов массива `data` начального значения 0; увеличение *k* на 1 для каждого элемента массива, не равного нулю; вывод результирующего значения переменной *k*.

Представим алгоритм блок-схемой:



Текст программы на языке Си будет следующим:

```
#include<stdio.h>
#include<stdlib.h>
#define n 7
void main()
{
   int a,b,i,data[n],k;
   scanf("%d%d",&a,&b);
   randomize();
   for(i=0;i<n;i++)
   {
      data[i]=a+random(b-a);
      printf("%4d",data[i]);
   }
   printf("\n");
   k=0;
   for(i=0;i<n;i++)
   {
      if(data[i]!=0)
      {
       k++;
      }
   }
   printf("%4d\n",k);
}
```

Результаты вычислений по программе:

Ввод		Вывод							
a	b	Массив data							k
-1	2	-1	2	2	0	1	-1	0	5
0	1	0	1	0	1	0	0	1	3

Лабораторная работа № 13

РАЗРАБОТКА И ПРОГРАММИРОВАНИЕ АЛГОРИТМА ОБРАБОТКИ ДВУМЕРНОГО МАССИВА

Цель: знакомство с понятием двумерного массива и приобретение навыков организации его обработки.

Задание. Разработать алгоритм обработки двумерного массива размера $n \times m$, заполненного целыми случайными числами из диапазона от a до b, в соответствии с поставленной задачей (табл. 13.1). Представить разработанный алгоритм в виде блок-схемы и программы для ЭВМ на алгоритмическом языке Си.

Таблица 13.1

Вариант	Задача обработки массива заключается в определении
1	произведения всех положительных элементов
2	произведения всех отрицательных элементов
3	произведения всех чётных элементов
4	произведения всех нечётных элементов
5	произведения всех кратных трём элементов
6	произведения всех не кратных трём элементов
7	количества всех положительных элементов
8	количества всех отрицательных элементов
9	количества всех чётных элементов
10	количества всех нечётных элементов
11	количества всех кратных трём элементов
12	количества всех не кратных трём элементов
13	минимального элемента
14	минимального элемента из всех положительных элементов
15	минимального элемента из всех чётных элементов
16	минимального элемента из всех нечётных элементов
17	минимального элемента из всех кратных трём элементов
18	минимального элемента из всех не кратных трём элементов

Продолжение табл. 13.1

Вариант	Задача обработки массива заключается в определении
19	максимального элемента
20	максимального элемента из всех отрицательных элементов
21	максимального элемента из всех чётных элементов
22	максимального элемента из всех нечётных элементов
23	максимального элемента из всех кратных трём элементов
24	максимального элемента из всех не кратных трём элементов
25	суммы всех положительных элементов
26	суммы всех отрицательных элементов
27	суммы всех чётных элементов
28	суммы всех нечётных элементов
29	суммы всех кратных трём элементов
30	суммы всех не кратных трём элементов

Основные положения

Двумерным массивом называется поименованная конечная последовательность одномерных массивов. Графически двумерный массив представляется прямоугольной таблицей. Размер двумерного массива обычно указывают в виде $n \times m$, где n – количество одномерных массивов (строк таблицы); m – количество элементов в каждом одномерном массиве (столбцов таблицы).

Разрабатываемый алгоритм должен содержать: ввод значений n, m, a и b; присвоение каждому элементу массива целого случайного числа из диапазона от a до b; дальнейшие действия, связанные с обработкой массива и выводом результата.

Двумерные массивы в языке Си описываются следующим образом:

```
тип имя_массива[число_строк][число_столбцов];
```

При этом компилятор отводит под массив память в количестве `(sizeof(тип) * число_строк * число_столбцов)` байтов, где `sizeof` является операцией, определяющей число байтов, занимаемой операндом `тип`.

Доступ к какому-либо элементу массива (ячейке таблицы) осуществляется посредством указания имени массива и двух номеров (индексов), первый из которых является номером одномерного массива (строки таблицы), а другой – номером элемента в этом одномерном массиве (столбца таблицы).

Двумерные массивы индексируются так же как и одномерные, начиная с нуля. Например, если целочисленный двумерный массив `data` описан как

```
int data[3][4];
```

то под него будет выделено `(sizeof(int) * 3 * 4)` = (2 * 12) = 24 байта, а двенадцатью элементами массива будут элементы: `data[0][0], data[0][1], data[0][2], data[0][3], data[1][0], data[1][1], data[1][2], data[1][3], data[2][0], data[2][1], data[2][2], data[2][3]`.

Для описания целочисленного двумерного массива `data`, состоящего, например, из 3 строк по 4 элемента в каждой строке, можно использовать инструкцию

```
int data[n][m];
```

с предварительным заданием значений *n* и *m* с помощью макроопределений

```
#define n 3
#define m 4
```

которые увязывают идентификаторы *n* и *m* со значениями 3 и 4 соответственно при любом появлении этих идентификаторов в тексте программы.

Под обработкой массива понимается анализ, замена, перестановка его элементов и другие возможные действия.

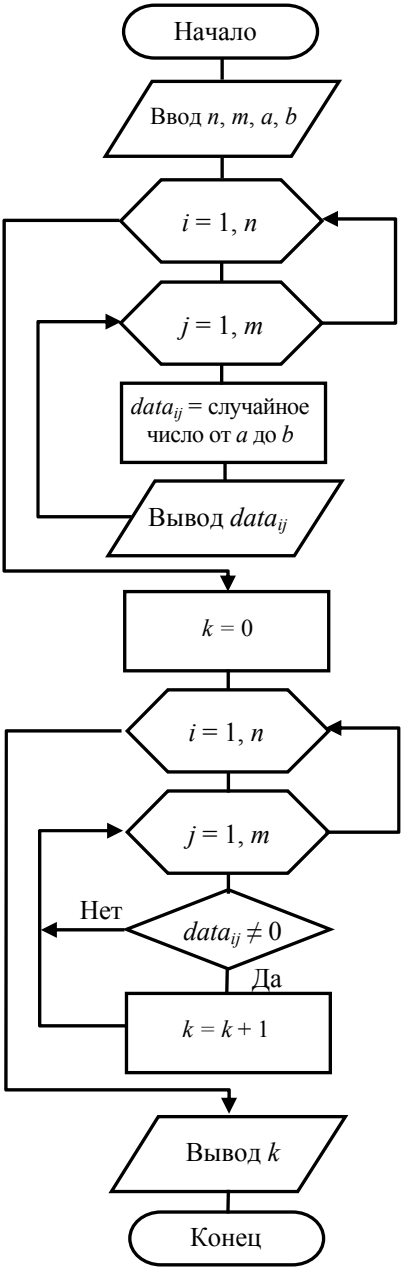
Пример выполнения задания

Задание. Разработать алгоритм обработки двумерного массива размера $n \times m$, заполненного целыми случайными числами из диапазона от *a* до *b*, в соответствии с поставленной задачей определения количества ненулевых элементов; представить алгоритм в виде блок-схемы и программы для ЭВМ на алгоритмическом языке Си.

Решение.

В разрабатываемом алгоритме предусмотрим: ввод значений *n*, *m*, *a* и *b*; присвоение каждому элементу двумерного массива (назовём его `data`) целого случайного числа из диапазона от *a* до *b*; вывод полученного массива data; присвоение количеству *k* ненулевых элементов массива `data` начального значения 0; увеличение *k* на 1 для каждого элемента массива, не равного нулю; вывод результирующего значения переменной *k*.

Представим алгоритм блок-схемой:



Текст программы на языке Си будет следующим:

```
#include<stdio.h>
#include<stdlib.h>
#define n 3
#define m 4
void main()
{
   int a,b,i,j,data[n][m],k;
   scanf("%d%d",&a,&b);
   randomize();
   for(i=0;i<n;i++)
   {
      for(j=0;j<m;j++)
      {
       data[i][j]=a+
             random(b-a+1);
       printf("%4d",
             data[i][j]);
      }
      printf("\n");
   }
   k=0;
   for(i=0;i<n;i++)
   for(j=0;j<m;j++)
   if(data[i][j]!=0)
   k++;
   printf("%4d\n",k);
}
```

Результаты вычислений по программе внесём в таблицу:

Ввод		Вывод				
a	b	Массив data				k
−1	2	0	−1	0	−1	10
		2	2	1	2	
		−1	−1	−1	2	
0	1	0	1	1	0	5
		0	0	1	0	
		0	1	1	0	

АЛГОРИТМ И ПРОГРАММА ОБРАБОТКИ СТРОКИ СИМВОЛОВ

Цель: знакомство с понятием строки символов и приобретение навыков организации обработки строки символов.

Задание. Разработать алгоритм обработки строки символов, которая может содержать буквы английского алфавита, цифры, знаки препинания, пробелы, знаки арифметических операций и скобки, в соответствии с поставленной задачей (табл. 14.1). Представить алгоритм в виде блок-схемы и программы для ЭВМ на алгоритмическом языке Си.

Таблица 14.1

Вариант	Задача обработки строки символов
1	Определение количества слов, разделённых пробелами, в заданном тексте
2	Проверка, является ли заданное слово палиндромом (одинаково читается как в прямом, так и в обратном направлении)
3	Определение, сколько раз в заданной строке символов встречается заданный символ
4	Проверка баланса открывающихся и закрывающихся скобок в заданной строке символов
5	Определение доли пробелов в заданной строке символов
6	Проверка, является ли заданное слово названием времени года
7	Замена всюду в заданной строке символов последовательности символов “`sin`” на последовательность символов “`cos`”
8	Определение общего количества точек и запятых в заданной строке символов
9	Удвоение каждого символа в заданной строке символов
10	Исключение всюду из заданной строки символов заданного символа
11	Определение, сколько раз в заданнной строке символов встречается заданная последовательность из двух символов
12	Присоединение к заданной строке символов строки символов, обратной по отношению к заданной строке
13	Определение, содержатся ли в заданной строке символов все символы другой заданной строки символов

Продолжение табл. 14.1

Вариант	Задача обработки строки символов
14	Определение количества предложений, заканчивающихся точкой или восклицательным знаком, в заданном тексте
15	Определение доли гласных букв в заданном слове
16	Исключение из заданной строки символов последовательностей символов, заключённых в кавычки
17	Вставка заданного символа после каждого символа в заданной строке символов
18	Проверка, является ли заданная строка символов обращением другой заданной строки символов
19	Проверка, имеется ли в заданной строке символов хотя бы одна пара соседних одинаковых символов
20	Исключение из заданной строки символов всех символов с чётными номерами
21	Определение, каких букв, гласных или согласных, больше в заданном слове
22	Определение встречаемости каждого символа в заданной строке символов
23	Определение количества арабских цифр в заданной строке символов
24	Замена в заданной строке символов каждой арабской цифры на следующую по порядку цифру (цифру 9 следует заменять на цифру 0)
25	Определение символов, каждый из которых встречается в заданной строке символов только один раз
26	Определение в заданном тексте количества слов, начинающихся с последовательности символов “`co`”
27	Определение длины самого короткого слова в заданном тексте
28	Определение в заданном тексте количества слов, заканчивающихся последовательностью символов “`ed`”
29	Исключение из заданной строки символов первого символа
30	Перестановка первого символа заданной строки символов в конец этой строки

Основные положения

Отдельный символ может занимать в памяти ЭВМ один байт и храниться в виде целого числа из диапазона от 0 до 255 включительно, называемого *кодом символа*. Таблицы, отражающие однозначные соответствия

Таблица 14.2

0 -	16 - ►	32 -	48 - 0	64 - @	80 - P	96 - `	112 - p
1 - ☺	17 - ◄	33 - !	49 - 1	65 - A	81 - Q	97 - a	113 - q
2 - ☻	18 - ↕	34 - «	50 - 2	66 - B	82 - R	98 - b	114 - r
3 - ♥	19 - ‼	35 - #	51 - 3	67 - C	83 - S	99 - c	115 - s
4 - ♦	20 - ¶	36 - $	52 - 4	68 - D	84 - T	100 - d	116 - t
5 - ♣	21 - §	37 - %	53 - 5	69 - E	85 - U	101 - e	117 - u
6 - ♠	22 - ▬	38 - &	54 - 6	70 - F	86 - V	102 - f	118 - v
7 - •	23 - ↨	39 - '	55 - 7	71 - G	87 - W	103 - g	119 - w
8 - ◘	24 - ↑	40 - (	56 - 8	72 - H	88 - X	104 - h	120 - x
9 - ○	25 - ↓	41 -)	57 - 9	73 - I	89 - Y	105 - i	121 - y
10 - ◙	26 - →	42 - *	58 - :	74 - J	90 - Z	106 - j	122 - z
11 - ♂	27 - ←	43 - +	59 - ;	75 - K	91 - [	107 - k	123 - {
12 - ♀	28 - ∟	44 - ,	60 - <	76 - L	92 - \	108 - l	124 - |
13 - ♪	29 - ↔	45 - -	61 - =	77 - M	93 -]	109 - m	125 - }
14 - ♫	30 - ▲	46 - .	62 - >	78 - N	94 - ^	110 - n	126 - ~
15 - ☼	31 - ▼	47 - /	63 - ?	79 - O	95 - _	111 - o	127 - ⌂

между кодами и символами, называются *таблицами кодировок*. Различные таблицы кодировок имеют общую часть с кодами от 0 до 127, совпадающую с кодировкой IBM [8]. Эта часть включает в себя латинские буквы, цифры, знаки пунктуации, знаки арифметических операций, скобки, символ пробел, а также другие символы и представлена в табл. 14.2.

Строкой символов называется некоторая последовательность символов, по сути, являющаяся одномерным массивом символов. В языке Си важной особенностью строк символов является то, что каждая из них заканчивается нулевым символом ‘\0’ [7]. Строка символов − константа ограничивается двойными кавычками. Например, описание строки символов и одновременно её инициализация могут быть выполнены следующим образом:

```
char str[50]= ”I love you.”
```

Размер этой строки − 12 байтов, несмотря на то, что сама строка содержит только 11 символов. В конец каждой строки компилятор подставляет признак конца строки символов − символ ‘\0’. Для нашего примера будем иметь следующую модель памяти:

I		l	o	v	e		y	o	u	.	\0

Таблица 14.3

`char *strcat(char *dest, char *source)` Конкатенирует строки `dest` и `source`.
`char *strchr(char *source, char ch)` Осуществляет поиск в строке `source` первое вхождение символа `ch`.
`int strcmp(char *s1, char *s2)` Сравнивает строки `s1` и `s2`. Возвращает `0`, если `s1 == s2`, возвращает `<0`, если `s1 < s2`, и возвращает `>0`, если `s1 > s2`.
`char *strcpy(char *dest, char *sourse)` Копирует строку `sourse` в строку `dest`.
`int strlen(char *s)` Определяет количество символов в строке `s` и возвращает это число.
`char *strrev(char *s)` Инвертирует все символы в строке `s`.

Для ввода строки символов `str` с клавиатуры можно использовать функцию `gets` с прототипом `char *gets(char *str)`, представленным в файле `stdio.h`.

В стандартной библиотеке языка Си содержатся полезные для пользователя функции для манипуляций со строками символов, некоторые из них представлены в табл. 14.3.

Заметим, что описание стандартных функций для работы со строками содержится в файле `string.h`, который следует подключать к тексту программы при использовании этих функций.

Обработка строки символов может осуществляться также путём доступа к отдельным символам строки как к элементам одномерного массива. Обратим при этом особое внимание на то, что если формирование некоторой строки символов в программе осуществляется посимвольно, то программист должен сам предусмотреть добавление в строку символа конца строки. Если же строка символов создаётся с помощью некоторой стандартной функции, то символ конца строки добавляется автоматически.

Разрабатываемый алгоритм должен содержать: ввод строки символов, подлежащей обработке, и другой необходимой информации (отдельных символов, строк символов); дальнейшие действия, связанные с обработкой заданной строки символов и выводом результата.

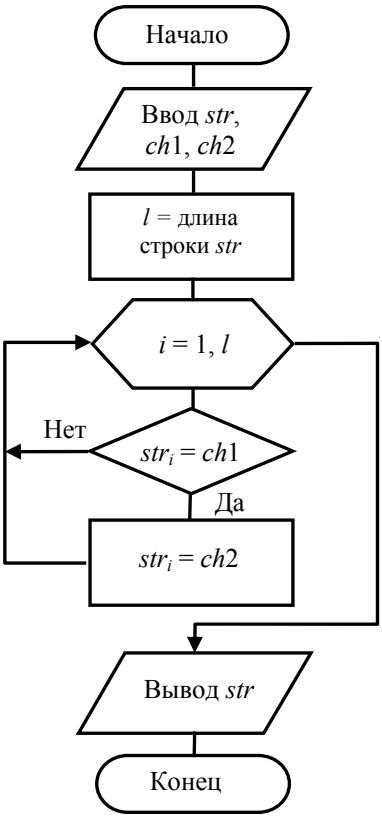
Пример выполнения задания

Задание. Разработать алгоритм обработки строки символов, которая может содержать буквы английского алфавита, цифры, знаки препинания, пробелы, знаки арифметических операций и скобки, в соответствии с поставленной задачей замены всюду в заданной строке символов одного заданного символа на другой заданный символ. Представить алгоритм в виде блок-схемы и программы для ЭВМ на алгоритмическом языке Си.

Решение.

В разрабатываемом алгоритме предусмотрим: ввод строки символов `str`, подлежащей обработке; ввод заменяемого `ch1` и заменяющего `ch2` символа; определение длины `l` строки `str`; сравнение в цикле каждого символа строки `str` с символом `ch1`, и в случае совпадения, необходимую замену символа; вывод полученной строки символов.

Представим алгоритм блок-схемой:



Текст программы на языке Си будет следующим:

```
#include<stdio.h>
#include<string.h>
void main()
{
   char str[100],ch1,ch2;
   int i,l;
   gets(str);
   scanf("%c %c",&ch1,&ch2);
   l=strlen(str);
   for(i=0;i<l;i++)
   {
      if(str[i]==ch1)
      str[i]=ch2;
   }
   printf("%s\n",str);
}
```

Приведём тестовые примеры для разработанной программы:

Ввод			Вывод
str	ch1	ch2	
campanent	a	o	component
unjversjty	j	i	university
a-b=c-d	-	+	a+b=c+d

Лабораторная работа № 15

ПРОГРАММИРОВАНИЕ ПОТОКОВОГО ВВОДА/ВЫВОДА

Цель: знакомство с понятием потокового ввода/вывода и приобретение навыков разработки программ с использованием ввода в файлы и вывода из файлов.

Задание. Разработать программу обработки файла, заполненного целыми случайными числами из диапазона от *a* до *b*, и помещения результатов в выходные файлы, в соответствии с поставленной задачей (табл. 15.1). Представить алгоритм в виде программы для ЭВМ на алгоритмическом языке Си.

Таблица 15.1

Вариант	Задача обработки файла
1	Записать в первый выходной файл сумму всех чётных элементов файла, а во второй − сумму всех нечётных элементов
2	Записать в выходной файл все не кратные трём, а затем все положительные элементы файла
3	Записать в выходной файл сначала все кратные трём, а затем все отрицательные элементы файла
4	Записать в первый выходной файл все кратные трём элементы файла, а во второй − все не кратные трём элементы
5	Записать в выходной файл сначала все не кратные трём, а затем все отрицательные элементы файла
6	Записать в выходной файл сначала все неотрицательные, а затем все чётные элементы файла
7	Записать в первый выходной файл все нечётные элементы файла, а во второй − все чётные элементы
8	Записать в выходной файл сначала все нечётные элементы, а затем положительные элементы файла
9	Записать в выходной файл сначала все нечётные, а затем все чётные элементы файла
10	Записать в первый выходной файл все отрицательные кратные трём элементы файла, а во второй − все отрицательные не кратные трём элементы
11	Записать в выходной файл сначала все отрицательные, а затем все чётные элементы файла
12	Записать в выходной файл сначала все отрицательные, а затем все нечётные элементы файла
13	Записать в первый выходной файл все отрицательные чётные элементы файла, а во второй − все отрицательные нечётные
14	Записать в выходной файл сначала все отрицательные, а затем все положительные элементы файла
15	Записать в выходной файл сначала все положительные, а затем все кратные трём элементы файла
16	Записать в первый выходной файл все положительные кратные трём элементы файла, а во второй − все положительные не кратные трём элементы
17	Записать в выходной файл сначала все элементы файла, кратные трём, а затем все элементы файла, не кратные трём

Продолжение табл. 15.1

Вариант	Задача обработки файла
18	Записать в выходной файл сначала количество отрицательных чётных элементов файла, а затем количество положительных нечётных элементов
19	Записать в первый выходной файл все положительные чётные элементы файла, а во второй − все отрицательные нечётные элементы
20	Записать в выходной файл сначала количество отрицательных элементов файла, а затем количество положительных элементов
21	Записать в выходной файл сначала минимальный элемент файла, а затем максимальный элемент
22	Записать в первый выходной файл количество нечётных элементов файла, а во второй − количество чётных элементов
23	Записать в выходной файл сначала сумму всех нечётных элементов файла, а затем сумму всех элементов, кратных трём
24	Записать в первый выходной файл максимальный из отрицательных элементов файла, а во второй − минимальный из положительных элементов
25	Записать в выходной файл сначала сумму всех положительных кратных трём элементов файла, а затем сумму всех отрицательных не кратных трём элементов
26	Записать в первый выходной файл сумму всех отрицательных кратных трём элементов файла, а во второй – сумму всех положительных не кратных трём элементов
27	Записать в выходной файл сначала сумму всех положительных нечётных элементов файла, а затем сумму всех положительных чётных элементов
28	Записать в первый выходной файл сумму всех отрицательных чётных элементов файла, а во второй − сумму всех положительных нечётных элементов
29	Записать в выходной файл сначала сумму всех отрицательных элементов файла, а затем сумму всех положительных элементов
30	Записать в первый выходной файл сумму всех элементов файла, не кратных трём, а во второй − сумму всех нечётных элементов

Основные положения

Понятие *поток* происходит из представления о последовательной структуре информационных записей [7]. Базовыми операциями над потоком являются следующие операции:

– считывание блока данных из потока в оперативной памяти (одна или несколько записей);

– запись блока данных из оперативной памяти в поток;

– обновление блока данных в потоке;

– занесение определённой записи данных в поток.

Состав потока задаётся структурой `FILE`, описание которой содержится в файле `stdio.h`.

Функция `fopen` используется для открытия потока (файла). Интерфейс с функцией `fopen` описывается следующим образом:

```
FILE *fopen(char *filename, char *type);
```

В качестве первого параметра `filename` должно передаваться правильное имя файла.

Второй параметр `type` определяет тип доступа к файлу, например:

`"r"` – открыть уже существующий файл на ввод;

`"w"` – создать новый файл или очистить уже существующий файл и открыть его на вывод;

`"a"` – создать новый файл для вывода или осуществить вывод в конец уже существующего файла;

`"r+"` – открыть существующий файл для обновления, которое будет проводиться с начала файла;

`"w+"` – создать новый файл или очистить существующий файл для обновления его содержимого;

`"a+"` – создать новый файл или подстроиться в конец существующего файла для обновления его содержимого.

Функция `fopen` возвращает указатель на структуру `FILE`, которая описывает файл.

Функция `fclose` используется для закрытия потока `stream`. Она имеет прототип

```
int fclose(FILE *stream);
```

Прототип функции `fread` выглядит следующим образом:

```
int fread(void *ptr, unsigned elem_size,
       int count, FILE *stream);
```

С помощью этой функции из входного потока `stream` считываются и по адресу `*ptr` записываются не более чем `count` элементов размером `elem_size` байтов каждый. Функция возвращает число фактически считанных элементов.

Функции `fwrite` имеет аналогичный прототип:

```
int fwrite(void *ptr, unsigned elem_size,
       int count, FILE *stream);
```

С помощью этой функции, начиная с адреса `*ptr`, считываются не более чем `count` элементов размером `elem_size` байтов каждый, и записываются в выходной поток `stream`. Функция возвращает число фактически записанных элементов.

Пример выполнения задания

Задание. Разработать программу для заполнения файла целыми случайными числами из диапазона от *a* до *b*, чтения данных из него и записи отрицательных чисел в первый, а положительных чисел − во второй выходные файлы. Представить алгоритм в виде программы для ЭВМ на алгоритмическом языке Си.

Решение.

Программа на языке Си будет иметь следующий вид с нумерацией строк исключительно для удобства её описания:

```
1)   #include<stdio.h>
2)   #include<stdlib.h>
3)   #include<time.h>
4)   #define n 7
5)   void main()
6)   {
7)      FILE  *in,*out_n,*out_p;
8)      int a,b,i,buf;
9)      scanf("%d%d",&a,&b);
10)     in=fopen("in_f","w");
11)     randomize();
12)     for(i=0;i<n;i++)
13)     {
14)        buf=a+random(b-a);
15)        fwrite(&buf,sizeof(int),1,in);
16)     }
17)     fclose(in);
```

```
18)     in=fopen("in_f","r");
19)     out_n=fopen("out_n_f","w");
20)     out_p=fopen("out_p_f","w");
21)     while(fread(&buf,sizeof(int),1,in)!=0)
22)     {
23)        printf("%4d",buf);
24)        if(buf<0)
25)        fwrite(&buf,sizeof(int),1,out_n);
26)        if(buf>0)
27)        fwrite(&buf,sizeof(int),1,out_p);
28)     }
29)     printf("\n");
30)     fclose(in);
31)     fclose(out_n);
32)     fclose(out_p);
33)     out_n=fopen("out_n_f","r");
34)     while(fread(&buf,sizeof(int),1,out_n)!=0)
35)     {
36)        printf("%4d",buf);
37)     }
38)     printf("\n");
39)     fclose(out_n);
40)     out_p=fopen("out_p_f","r");
41)     while(fread(&buf,sizeof(int),1,out_p)!=0)
42)     {
43)        printf("%4d",buf);
44)     }
45)     printf("\n");
46)     fclose(out_p);
47)  }
```

Строка 3 программы предназначена для подключения к тексту файла `time.h` для корректной работы функции `randomize`.

В программе `n` – количество целых чисел во входном файле; `in`, `out_n`,`out_p` – указатели на входной файл и выходные файлы для отрицательных и положительных чисел; `i` – параметр цикла; `buf` – переменная для временного хранения считываемых и записываемых целых чисел.

В строке 10 открывается файл с именем `in_f` на запись, в цикле с 12 по 16 строки этот файл заполняется целыми случайными числами из диапазона от `a` до `b` и в строке 17 файл закрывается.

В строках с 18 по 20 файл `in_f` открывается на чтение и файлы `out_n_f`, `out_p_f` – на запись.

В цикле с 21 по 28 строки, пока из входного потока `in` считываются блоки размером `sizeof(int)` байтов, выводится на экран прочитанное значение, которое затем записывается в выходной поток `out_n`, если оно отрицательное, и в выходной поток `out_p`, если положительное.

В строках с 30 по 32 закрываются потоки `in, out_n` и `out_p`.

В строках с 33 по 39 считываются и выводятся на экран монитора данные из файла `out_n_f`, а в строках с 40 по 46 – из файла `out_p_f`.

Результаты работы программы внесём в таблицу:

Ввод		Вывод		
a	b	in_f	out_n_f	out_p_f
-3	4	-3 2 0 2 -3 3 0	-3 -3	2 2 3
-4	3	2 -4 -2 -3 1 0 -1	-4 -2 -3 -1	2 1
-2	5	4 1 0 -2 -1 3 1	-2 -1	4 1 3 1

Лабораторная работа № 16

КОД ХЕММИНГА

Цель: знакомство с понятием кода Хемминга и приобретение навыков его применения для обнаружения и коррекции одиночных ошибок, возникающих при хранении информации.

Задание.

1. Для заданного 16-разрядного слова данных из табл. 16.1 получить соответствующий код Хемминга.

2. По заданному в табл. 16.1 коду Хемминга с одиночной ошибкой обнаружить место этой ошибки и исправить её.

Основные положения

При работе современных запоминающих устройств, хранящих информацию в двоичном виде, могут возникать ошибки, обусловленные воздействием различных факторов. Для повышения надёжности работы этих устройств используют специальные коды [1].

Различают коды, которые *обнаруживают ошибки* и *корректирующие коды*. Простейшим способом обнаружения ошибок является *проверка на чётность*. В этом случае к битам хранимого M-разрядного слова добавляют ещё один бит – бит чётности, значение которого подбирается таким образом, чтобы среди получившихся $N = M + 1$ разрядов обязательно было чётное число единиц. Такой избыточный код позволяет лишь констатировать факт наличия ошибки.

Таблица 16.1

Вари-ант	16-разрядное слово данных	Код Хемминга с ошибкой
1	1010011011011001	011001111011001100010
2	0101000001110010	100100110011010010110
3	0111001011001000	010100100001001001001
4	1100010100000010	100110000110010111001
5	1111100110000100	100011000001011100010
6	0111101111101010	100110001100000111100
7	0110111100001000	111001111010110111111
8	0011101100101000	011001011010100001010
9	1011011110001001	000111100110001010111
10	1101100010001000	011100101101001011110
11	1010010010100101	001111111110111110000
12	1000101111110111	101100100111010011001
13	0110101111000001	100011110001110110111
14	0000110111001010	010111010000100110111
15	0101111100101110	110011011100100110001
16	0100110100111110	111110100010011001111
17	0110111001101100	111100001011000110011
18	1010001000101010	001010010110001110100
19	0011011100101011	010100011111100111011
20	0110111101111101	001100101001111011000
21	1100111101010011	101010000010111101011
22	1010101111110100	010101111111001010111
23	0111110101110111	110100001110010011100
24	0000000001000111	010010100001010010110
25	1100000101000011	110011010110111011001
26	1101101110111101	100011000110100110000
27	1111100010001100	100010110000111100111
28	1100000100101101	010100000110100010001
29	0000100010010111	111010101101100111010
30	1000000111111100	100000100111011101110

Корректирующие коды основаны на том, что к каждому хранимому *M*-разрядному слову добавляют *K* битов чётности с соответствующим их расположением между битами этого слова. Подобные *N*-разрядные коды ($N = M + K$) были впервые рассмотрены американским математиком Ричардом Хеммингом (1915 – 1998) и впоследствии были названы его именем.

Для заданного *M* число разрядов *N* кода Хемминга определяется соотношением

$$2^{N-M} - 1 \geq N .$$

Все биты *N*-разрядного кода Хемминга нумеруются слева направо, начиная с 1. При этом биты чётности имеют номера, равные некоторой степени числа 2, а остальные биты являются информационными.

Каждый бит чётности используется для контроля определённых информационных разрядов *N*-разрядного слова.

Первый бит чётности контролирует разряды кода Хемминга с номерами $a_1 \times 2^1 + 1 \times 2^0$, где $a_1 = 0, \ldots, N / 2^1$.

Второй бит чётности контролирует разряды кода Хемминга с номерами $a_2 \times 2^2 + 1 \times 2^1 + a_0 \times 2^0$, где $a_2 = 0, \ldots, N / 2^2$; $a_0 = 0, 1$.

Третий бит чётности контролирует разряды кода Хемминга с номерами $a_3 \times 2^3 + 1 \times 2^2 + a_1 \times 2^1 + a_0 \times 2^0$, где $a_3 = 0, \ldots, N / 2^3$; $a_1 = 0, 1$; $a_0 = 0, 1$.

Четвёртый и последующие биты чётности определяют контролируемые разряды аналогичным образом.

К примеру, номера контролируемых разрядов для первых пяти битов чётности при $N = 31$ приведены в табл. 16.2.

При этом содержимое бита чётности устанавливается так, чтобы суммарное число единиц в контролируемых разрядах было чётным.

Если из-за воздействия каких-либо возмущающих факторов произойдёт изменение в одном из разрядов кода Хемминга, то номера разря-

Таблица 16.2

Бит чётности / Разряд кода Хемминга	Контролируемые разряды
1 / 1	1, 3, 5, 7, 9, 11, 13, 15, 17, 19, 21, 23, 25, 27, 29, 31
2 / 2	2, 3, 6, 7, 10, 11, 14, 15, 18, 19, 22, 23, 26, 27, 30, 31
3 / 4	4, 5, 6, 7, 12, 13, 14, 15, 20, 21, 22, 23, 28, 29, 30, 31
4 / 8	8, 9, 10, 11, 12, 13, 14, 15, 24, 25, 26, 27, 28, 29, 30, 31
5 / 16	16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31

дов соответствующих битам чётности, для которых при этом нарушены контрольные суммы, позволяют установить место ошибки, а, следовательно, и исправить её.

Пример выполнения задания

Задание 1. Для заданного 16-разрядного слова данных 0110100000010011 получить соответствующий код Хемминга.

Решение.

Для заданного числа информационных разрядов $M = 16$ определим число разрядов N кода Хемминга с помощью соотношения $2^{N-M} - 1 \geq N$:

$$\left.\begin{array}{l}(2^{20-16} - 1 \geq 20) = (2^4 - 1 \geq 20) = (15 \geq 20) = \text{Л} \\ (2^{21-16} - 1 \geq 21) = (2^5 - 1 \geq 21) = (31 \geq 21) = \text{И}\end{array}\right\} \Rightarrow N = 21.$$

Вычислим количество битов чётности $K = N - M = 21 - 16 = 5$.

Все биты 21-разрядного кода Хемминга пронумеруем слева направо, начиная с 1. При этом биты чётности имеют номера 1, 2, 4, 8 и 16, равные степеням 0, 1, 2, 3 и 4 числа 2, а остальные биты с номерами 3, 5, 6, 7, 9, 10, 11, 12, 13, 14, 15, 17, 18, 19, 20 и 21 являются информационными.

Заполним информационные биты кода Хемминга цифрами заданного 16-разрядного слова 0110100000010011.

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

Определим необходимые значения битов чётности кода Хемминга.

1-й бит) Сумма «3» + «5» + «7» + «9» + «11» + «13» + «15» + «17» + + «19» + «21» = 0 + 1 + 0 + 1 + 0 + 0 + 0 + 1 + 0 + 1 = 4. Сумма является четным числом, поэтому в первый бит чётности установим 0.

2-й бит) Сумма «3» + «6» + «7» + «10» + «11» + «14» + «15» + «18» + + «19» = 0 + 1 + 0 + 0 + 0 + 0 + 0 + 0 + 0 = 1. Сумма является нечётным числом, поэтому во второй бит чётности установим 1.

3-й бит) Сумма «5» + «6» + «7» + «12» + «13» + «14» + «15» + «20» + + «21» = 1 + 1 + 0 + 0 + 0 + 0 + 0 + 1 + 1 = 4. Сумма является чётным числом, поэтому в третий бит чётности установим 0.

4-й бит) Сумма «9» + «10» + «11» + «12» + «13» + «14» + «15» = 1 + 0 + + 0 + 0 + 0 + 0 + 0 = 1. Сумма является нечётным числом, поэтому в четвертый бит чётности установим 1.

5-й бит) Сумма «17» + «18» + «19» + «20» + «21» = 1 + 0 + 0 + 1 + 1 = 3. Сумма является нечётным числом, поэтому в пятый бит чётности установим 1.

Дополним информационные биты битами чётности и получим искомый код Хемминга.

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

Ответ: 010011011000000110011.

Задание 2. По заданному коду Хемминга с одиночной ошибкой 101000000110111101101 обнаружить место этой ошибки и исправить её.

Решение.

Заданный код Хемминга 101000000110111101101 имеет 21 разряд, следовательно, $N = 21$.

Для установленного числа разрядов кода Хемминга $N = 21$ определим число его информационных разрядов M с помощью соотношения $2^{N-M} - 1 \geq N$:

$$\left.\begin{array}{l}(2^{21-16} - 1 \geq 21) = (2^5 - 1 \geq 20) = (31 \geq 21) = \text{И} \\ (2^{21-17} - 1 \geq 21) = (2^4 - 1 \geq 21) = (15 \geq 21) = \text{Л}\end{array}\right\} \Rightarrow M = 16 .$$

Таким образом, количество битов чётности $K = N - M = 21 - 16 = 5$.

Все биты заданного 21-разрядного кода Хемминга с одиночной ошибкой пронумеруем слева направо, начиная с 1.

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

При этом биты с номерами 1, 2, 4, 8 и 16, равными степеням 0, 1, 2, 3 и 4 числа 2, являются битами чётности, а остальные с номерами 3, 5, 6, 7, 9, 10, 11, 12, 13, 14, 15, 17, 18, 19, 20 и 21 являются информационными.

Проверим на чётность контрольные суммы для каждого из пяти битов чётности кода Хемминга с ошибкой.

Для первого бита чётности контрольная сумма «1» + «3» + «5» + «7» + + «9» + «11» + «13» + «15» + «17» + «19» + «21» = 1 + 1 + 0 + 0 + 0 + 1 + 1 + + 1 + 0 + 1 + 1 = 7. Сумма является нечетным числом, что сигнализирует о наличии ошибки в одном из битов: 1, 3, 5, 7, 9, 11, 13, 15, 17, 19, 21.

Для второго бита четности контрольная сумма «2» + «3» + «6» + «7» + + «10» + «11» + «14» + «15» + «18» + «19» = 0 + 1 + 0 + 0 + 1 + 1 + 1 + 1 + + 1 + 1 = 7. Сумма является нечётным числом, что сигнализирует о наличии ошибки в одном из битов: 2, 3, 6, 7, 10, 11, 14, 15, 18, 19.

Для третьего бита чётности контрольная сумма «4» + «5» + «6» + «7» + + «12» + «13» + «14» + «15» + «20» + «21» = 0 + 0 + 0 + 0 + 0 + 1 + 1 + 1 + + 0 + 1 = 4. Поскольку сумма является чётным числом, каждый из битов 4, 5, 6, 7, 12, 13, 14, 15, 20, 21 правилен.

Для четвёртого бита чётности контрольная сумма «8» + «9» + «10» + + «11» + «12» + «13» + «14» + «15» = 0 + 0 + 1 + 1 + 0 + 1 + 1 + 1 = 5. Сумма является нечётным числом, что сигнализирует о наличии ошибки в одном из битов: 8, 9, 10, 11, 12, 13, 14, 15.

Наконец, для пятого бита чётности контрольная сумма «16» + «17» + + «18» + «19» + «20» + «21» = 1 + 0 + 1 + 1 + 0 + 1 = 4. Следовательно, каждый из битов 16, 17, 18, 19, 20, 21 правилен.

Вычислим номер искажённого бита как сумму номеров первого, второго и четвёртого битов чётности, для которых получены нечётные контрольные суммы: 1 + 2 + 8 = 11.

Исправим ошибку, изменив значение одиннадцатого разряда кода Хемминга с 1 на 0: 10100000011011110 1101 → 10100000010011110 1101.

Ответ: ошибка обнаружена в разряде 11, скорректированный код Хемминга равен 10100000010011110 1101.

Лабораторная работа № 17

КРИПТОАНАЛИЗ ШИФРА СДВИГА

Цель: знакомство с понятиями криптографии и криптоанализа, а также приобретение навыков криптоанализа шифра сдвига.

Задание. Расшифровать криптограмму (табл. 17.1), полученную из открытого текста с помощью шифра сдвига.

Основные положения

Криптографией (от греч. κρυπτός – скрытый и γράφω – пишу) называют науку о методах обеспечения конфиденциальности (невозможности прочтения посторонним) информации путём её шифрования.

Алгоритм шифрования (или *шифр*) − это преобразование открытого текста в зашифрованный (или *шифротекст*, *шифрограмму*, *криптограмму*) с помощью секретного ключа [9]. Этот процесс обозначают следующим образом:

$$C = E_k(m),$$

где m – *открытый текст*; E – *шифрующая функция*; k – *секретный ключ* и C – *шифротекст*.

Таблица 17.1

Вариант	Криптограмма
1	Э ЯЙЗА ЬАВ БГЖЧСЙЭ ГВЭАМНИЦР ИЫМАЕЙЗЦР ИА ЙЬЛЪФАУЧ
2	НЩЯЮБЬ ЮВХПЯЗ Ц БВЕ, ВБФ АФЙБЩЖ ЙЦВЕЖВА
3	ЖЮ ЙЗЪЮВ ИЮЙЮЭ ЫЙЩЬЗЕ, ДЧЛЮВСБВ ЫЙЩЬ РЮДЗЫЮГЩ ЗЖ КЩЕ
4	ИЖРЧЖ ЭЛЦУОШ Ч ХФСВНФД, Ж НСФП ЭЛСФИЛР Ч ЩКФИФСВЧШИОЛТ
5	Б КЩДЗ, МИЗЛЙЮЪДЮЖЖЗЮ ЖЮ Ы ЕЮЙМ, ЕЗЯЮЛ ИЙБРБЖБЛХ ЫЙЮЭ
6	ЧЯГДФШ ЦМТРЪШЯ ЦМТСЮЭЛ ЮЪ, Ц ГСШЯ ЪЩ ФШССЮ ЪБЪЮЯ
7	КЯКМКФЬЙЙЧЕ НПАШЭКЪ, ОЧ ЮНБ В ЙБ КОУЬДЮЬЕНЫ
8	РЖПРЕР БЛШВ ЖДВ ТВЙВ ПЗ ДЭУКЖКЪЮ
9	ФЯЮЩ ЫПТЯУН ЖЯГН У ХУЦБН, ЯЮС УЬЦГЩГ У ЯЫЮЯ
10	Р ЯОЬЗЛАЛ МЛНПКЛАЛ ОЯЛЖ ЯДАИЬБ КЭ ЕОЗРООПЯЛ
11	КВ ЯОЬЗЛЙР ЛСЕУВНР ЙРКБЕН З ИЕУР
12	ЫЩЬНСЪЕ, УЙУ ХЬЩЙЛОТ, ОЪФС ЮЧАОБЕ КДЫЕ ЬШЧНЧКФОЦ ШАОФО
13	ДРЗППЭЗ НАЖК ЙВЫКЫВАФ РФЗЩЗУФДР
14	ГЭЫПЭ ЬЭИЮШКФ ЪЖЮЮА, ПЭД ЩИШЯЬУ ЗИШЪГЭЕАЧ
15	В ЖНЮКХГ ЬИЕЦМЯК ЛИЖЗЯЬЪЕЛЩ Ь ЩЮИЬВМИЛМВ ДИОЯ
16	ФЗКС КФСПЦОМЧ, ЦЧ ЦО НЙЛЙТЪИ ОХЬ Л ЧКХЙЦ
17	ЬЕГВЯБЩБЬЩ ГДЩШГДЬУЖЬУ ГДЬУЖБВ НЩЮВЛЩЖ ЕФАВЯТХЬЩ
18	ВЪ ЧЖФЯХФ ОЪЯГЗЯХ ЩГЖЗХЧАФЪЗ ИЩГЧГАСЖЗЧЭЪ

Продолжение табл. 17.1

Вариант	Криптограмма
19	ЩДШДЖХ З ЛЮИЖЫМДВ, ШЭШЫЮЮШЦЯ ДИШЫИ ЗШДЯ
20	ЮЦ УЯ УВРЫЯЪ ЩФБЦ ГДШМ УМЩФБМУСПГ
21	ВЛЗА ЗИЭЮЬЭ ВШИКАЕЛ, Ш ЗЖЙГЭ ИШДВЛ
22	УПВКЕ ЙКН АМПЯДИ НКЭЗЬГЙ
23	РЪПТЭЖФАБЧФ Ч АВФСФЯЧФ - УСФ ЯПЦЬЧЕК
24	ФБЧЯЬЛФБЬБ БЩ ЯТХЬЖ АУЕФ, ЮВЖВДВЩ БЩ ЦГВЯЕПДВ
25	ОТФ ЪНСЫОЪЫЮЯХ ЪЫЮХЩИЦ ЪНОЭЛЕЪХЧ ПЭТСТЪ
26	АДНЛЪ ФГЧЮЕ ЯТЙТЭА Ъ ДН ЮЯАХАЧ БАЫЮЧКО
27	ЖЗКИИЯЦ ЫЕВЭДЧ ШТЙУ БЕЗЕВЬЩИЙЩЕГ
28	ЭЬЩЦДЦН Р ФЦХЫЦ ШОФТЬСЬ СЬЯБТОЮЯАРО УЯАК
29	ЦЧЪШЗС МШЩГ ХЛПФ ПО ФЗПИХТММ ФЗОХРТПЙВЬ ФЗШМСХУВЬ
30	М ЫЩПЪЬШЦ МШСОЭЯП ЩЪТ МЫПЦ ЫЬКЪКЧТТ ЧП ШЬОЕВТВЖЫЙ

Обратный по отношению к шифрованию процесс называют *расшифрованием* и пишут

$$m = D_k(C),$$

где D – *дешифрующая функция*.

Заметим, что функции шифрования и дешифрования E и D открыты, а секретность исходного текста m зависит от секретности ключа k. Как прямой процесс шифрования, так и обратный процесс расшифрования используют один и тот же ключ, в связи с чем такие алгоритмы в совокупности принято называть *симметричными криптосистемами* или *криптосистемами с секретным ключом*.

Существуют также криптосистемы, зависящие от двух различных ключей. Первый из них открыт и используется для шифрования, а второй (секретный) − при восстановлении открытого текста из шифрограммы.

Таблица 17.2

А а 0	Б б 1	В в 2	Г г 3	Д д 4	Е е (Ё ё) 5	Ж ж 6	З з 7
И и 8	Й й 9	К к 10	Л л 11	М м 12	Н н 13	О о 14	П п 15
Р р 16	С с 17	Т т 18	У у 19	Ф ф 20	Х х 21	Ц ц 22	Ч ч 23
Ш ш 24	Щ щ 25	Ъ ъ 26	Ы ы 27	Ь ь 28	Э э 29	Ю ю 30	Я я 31

Такие криптосистемы называют *асимметричными* или *криптосистемами с открытым ключом*.

Криптоанализ – наука о методах раскрытия зашифрованной информации без заранее известного ключа.

Познакомимся с одним из первых известных шифров, применявшимся для защиты информации в докомпьютерную эпоху и называемым *шифром сдвига*. В нём процесс шифрования заключается в замене каждой буквы открытого текста на другую, которая отстоит от исходной на определённое число позиций в алфавите в зависимости от значения ключа. Так, например, если ключ равен трём, то буква «а» исходного текста в криптограмме представляется буквой «Г», буква «б» исходного текста в криптограмме представляется буквой «Д» и т.д. При этом буквы «е» и «ё» часто отождествляются. Заметим, что если шифр сдвига используется с ключом, равным именно трём, его называют *шифром Цезаря*.

Математическое описание шифра сдвига основывается на том, что все буквы русского алфавита нумеруются, начиная с нуля. Например, букве «А а» присваивается номер 0, букве «Б б» − номер 1, ..., буквам «Е е» и «Ё ё» − номер 5, ..., букве «Я я» − номер 31 (табл. 17.2).

При шифровании к номеру x буквы открытого текста прибавляют значение ключа k по модулю 32 и в результате получают номер y соответствующей буквы криптограммы:

$y = x + k \pmod{32}$ = остаток от целочисленного деления $(x + k)$ на 32.

Заметим, что для представления открытого текста принято использовать строчные буквы, а для зашифрованного – прописные.

Например, если ключ k шифра сдвига равен 17, слово «криптография» преобразует в криптограмму «ЫБЩАГЯФБСЕЩР» следующим образом:

«к» + k (mod 32) = 10 + k (mod 32) = 10 + 17 (mod 32) = 27 (mod 32) = = 27 = «Ы»

«р» + k (mod 32) = 16 + k (mod 32) = 16 + 17 (mod 32) = 33 (mod 32) = = 1 = «Б»

«и» + k (mod 32) = 8 + k (mod 32) = 8 + 17 (mod 32) = 25 (mod 32) = = 25 = «Щ»

«п» + k (mod 32) = 15 + k (mod 32) = 15 + 17 (mod 32) = 32 (mod 32) = = 0 = «А»

«т» + k (mod 32) = 18 + k (mod 32) = 18 + 17 (mod 32) = 35 (mod 32) = = 3 = «Г»

«о» + k (mod 32) = 14 + k (mod 32) = 14 + 17 (mod 32) = 31 (mod 32) = = 31 = «Я»

«г» + k (mod 32) = 3 + k (mod 32) = 3 + 17 (mod 32) = 20 (mod 32) = = 20 = «Ф»

«р» + k (mod 32) = 16 + k (mod 32) = 16 + 17 (mod 32) = 33 (mod 32) = = 1 = «Б»

«а» + k (mod 32) = 0 + k (mod 32) = 0 + 17 (mod 32) = 17 (mod 32) = = 17 = «С»

«ф» + k (mod 32) = 20 + k (mod 32) = 20 + 17 (mod 32) = 37 (mod 32) = = 5 = «Е»

«и» + k (mod 32) = 8 + k (mod 32) = 8 + 17 (mod 32) = 25 (mod 32) = = 25 = «Щ»

«я» + k (mod 32) = 31 + k (mod 32) = 31 + 17 (mod 32) = 48 (mod 32) = = 16 = «Р»

Для расшифрования от номера y буквы шифротекста отнимают значение ключа k по модулю 32 и в результате получают номер x соответствующей буквы открытого текста:

$$x = y - k \text{ (mod 32)} = \begin{cases} y - k, \text{ если } y - k \geq 0; \\ y - k + 32, \text{ в противном случае.} \end{cases}$$

Например, если при шифровании ключ k шифра сдвига был равен 17, криптограмма «ЫБЩАГЯФБСЕЩР» преобразуется в слово «криптография» следующим образом:

«Ы» – k (mod 32) = 27 – k (mod 32) = 27 – 17 (mod 32) = 10 (mod 32) = = 10 = «к»

«Б» – k (mod 32) = 1 – k (mod 32) = 1 – 17 (mod 32) = – 16 (mod 32) = = – 16 + 32 = 16 = «р»

«Щ» – k (mod 32) = 25 – k (mod 32) = 25 – 17 (mod 32) = 8 (mod 32) = = 8 = «и»
«А» – k (mod 32) = 0 – k (mod 32) = 0 – 17 (mod 32) = – 17 (mod 32) = = – 17 + 32 = 15 = «п»
«Г» – k (mod 32) = 3 – k (mod 32) = 3 – 17 (mod 32) = – 14 (mod 32) = = – 14 + 32 = 18 = «т»
«Я» – k (mod 32) = 31 – k (mod 32) = 31 – 17 (mod 32) = 14 (mod 32) = = 14 = «о»
«Ф» – k (mod 32) = 20 – k (mod 32) = 20 – 17 (mod 32) = 3 (mod 32) = = 3 = «г»
«Б» – k (mod 32) = 1 – k (mod 32) = 1 – 17 (mod 32) = – 16 (mod 32) = = – 16 + 32 = 16 = «р»
«С» – k (mod 32) = 17 – k (mod 32) = 17 – 17 (mod 32) = 0 (mod 32) = = 0 = «а»
«Е» – k (mod 32) = 5 – k (mod 32) = 5 – 17 (mod 32) = – 12 (mod 32) = = – 12 + 32 = 20 = «ф»
«Щ» – k (mod 32) = 25 – k (mod 32) = 25 – 17 (mod 32) = 8 (mod 32) = = 8 = «и»
«Р» – k (mod 32) = 16 – k (mod 32) = 16 – 17 (mod 32) = – 1 (mod 32) = = – 1 + 32 = 31 = «я»

Криптостойкость шифра сдвига крайне низка. Наивный путь атаки на этот шифр состоит в простом переборе возможных значений ключа до тех пор, пока не получится осмысленный текст. Поскольку существует ровно 31 вариант таких значений (ключ со значением 0 не изменяет текст), то для расшифрования криптограммы потребуется не очень много времени.

Другой путь взлома шифра сдвига опирается на статистику используемого языка. В таблице 17.3 представлены частоты (в порядке убывания) появления в тексте на русском языке букв алфавита, в котором отождествлены «е» с «ё» и «ь» с «ъ», а также имеется символ пробела «–» между словами [10].

Таблица 17.3

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

Пример выполнения задания

Задание. Расшифровать криптограмму «ТУЗАК ЦЕФУИН ЗЦКИЙЕ ЛСШЧ», полученную из открытого текста с помощью шифра сдвига.

Решение.

Не будем производить наивную атаку на шифр сдвига вследствие её трудоемкости, а применим подход, основанный на статистике используемого (русского) языка.

Рассмотрим заданную криптограмму «ТУЗАК ЦЕФУИН ЗЦКИЙЕ ЛСШЧ» с точки зрения встречаемости букв русского алфавита. В ней, например, буква «А» встречается только один раз, буквы «Б», «В», «Г» и «Д» не встречаются ни разу. Буква «Е» встречается 2 раза, «Ж» не встречается, «З» встречается 2 раза и т.д. Определим сколько раз встречаются и остальные буквы, присутствующие в криптограмме, и поместим полученные данные в таблицу.

А	Е	З	И	Й	К	Л	Н	С	Т	У	Ф	Ц	Ч	Ш
1	2	2	2	1	2	1	1	1	1	2	1	2	1	1

Как видно из этой таблицы, в криптограмме наибольшее число раз, равное двум, встречаются буквы: «Е», «З», «И», «К», «У» и «Ц». В то же время статистика используемого русского языка, приведённая в табл. 17.3, свидетельствует о том, что наибольшую частоту появления в тексте имеет буква «о».

Поэтому логично предположить, что буква «о» открытого текста в криптограмме скрывается под видом одной из букв: «Е», «З», «И», «К», «У» или «Ц».

Будем считать вначале, что буква «Е» криптограммы представляет букву «о» открытого текста. Тогда сдвиг

$$k = y - x \pmod{32} = \text{«Е»} - \text{«о»} \pmod{32} = 5 - 14 \pmod{32} =$$

$$= -9 \pmod{32} = -9 + 32 = 23$$

и криптограмма преобразуются в бессмысленный текст «ыьрйу …»:

$$\text{«Т»} - k \pmod{32} = 18 - 23 \pmod{32} = -5 \pmod{32} = -5 + 32 = 27 = \text{«ы»}$$

$$\text{«У»} - k \pmod{32} = 19 - 23 \pmod{32} = -4 \pmod{32} = -4 + 32 = 28 = \text{«ь»}$$

$$\text{«З»} - k \pmod{32} = 7 - 23 \pmod{32} = -16 \pmod{32} = -16 + 32 = 16 = \text{«р»}$$

«А» – k (mod 32) = 0 – 23 (mod 32) = – 23 (mod 32) = – 23 + 32 = 9 = «й»

«К» – k (mod 32) = 10 – 23 (mod 32) = – 13 (mod 32) = – 13 + 32 = = 19 = «у»

. . .

Затем предположим, что буква «З» криптограммы представляет букву «о» открытого текста. Тогда сдвиг равен 25 и криптограмма вновь преобразуется в бессмысленный открытый текст «щъозс ...». Аналогичные результаты дают и предположения о том, что буквы «И», «К» представляют наиболее встречаемую букву «о».

Наконец, предположим, что буква «о» открытого текста скрывается под видом буквы «У». Следовательно, сдвиг при шифровании был равен 5:

$$k = y - x \text{ (mod 32)} = \text{«У» – «о» (mod 32)} = 19 - 14 \text{ (mod 32)} = 5 \text{ (mod 32)} = 5,$$

при этом из криптограммы получаем осмысленный открытый текст:

«Т» – k (mod 32) = 18 – 5 (mod 32) = 13 (mod 32) = 13 = «н»

«У» – k (mod 32) = 19 – 5 (mod 32) = 14 (mod 32) = 14 = «о»

«З» – k (mod 32) = 7 – 5 (mod 32) = 2 (mod 32) = 2 = «в»

«А» – k (mod 32) = 0 – 5 (mod 32) = – 5 (mod 32) = – 5 + 32 = 27 = «ы»

«К» – k (mod 32) = 10 – 5 (mod 32) = 5 (mod 32) = 5 = «е»

«Ц» – k (mod 32) = 22 – 5 (mod 32) = 17 (mod 32) = 17 = «с»

«Е» – k (mod 32) = 5 – 5 (mod 32) = 0 (mod 32) = 0 = «а»

«Ф» – k (mod 32) = 20 – 5 (mod 32) = 15 (mod 32) = 15 = «п»

«У» – k (mod 32) = 19 – 5 (mod 32) = 14 (mod 32) = 14 = «о»

«И» – k (mod 32) = 8 – 5 (mod 32) = 3 (mod 32) = 3 = «г»

«Н» – k (mod 32) = 13 – 5 (mod 32) = 8 (mod 32) = 8 = «и»

«З» – k (mod 32) = 7 – 5 (mod 32) = 2 (mod 32) = 2 = «в»

«Ц» – k (mod 32) = 22 – 5 (mod 32) = 17 (mod 32) = 17 = «с»

«К» – k (mod 32) = 10 – 5 (mod 32) = 5 (mod 32) = 5 = «е»

«И» – k (mod 32) = 8 – 5 (mod 32) = 3 (mod 32) = 3 = «г»

«Й» – k (mod 32) = 9 – 5 (mod 32) = 4 (mod 32) = 4 = «д»

«Е» – k (mod 32) = 5 – 5 (mod 32) = 0 (mod 32) = 0 = «а»

«Л» – k (mod 32) = 11 – 5 (mod 32) = 6 (mod 32) = 6 = «ж»

«С» – k (mod 32) = 17 – 5 (mod 32) = 12 (mod 32) = 12 = «м»

«Ш» – k (mod 32) = 24 – 5 (mod 32) = 19 (mod 32) = 19 = «у»

«Ч» – k (mod 32) = 23 – 5 (mod 32) = 18 (mod 32) = 18 = «т»

Ответ: «новые сапоги всегда жмут».

ЗАКЛЮЧЕНИЕ

Современное общество живёт в период небывалого роста объёма информационных потоков как в экономике, так и в социальной сфере. В промышленности рост объёма информации обусловлен увеличением производства, усложнением выпускаемой продукции, расширением внешних и внутренних связей экономических объектов.

К материальным, трудовым, энергетическим и финансовым ресурсам добавился информационный ресурс. Только на основе своевременного получения, накопления и переработки информационного ресурса возможно рациональное управление любой сферой человеческой деятельности и принятие правильных решений. Применение современных ЭВМ даёт возможность переложить трудоёмкие вычислительные и логические операции с человека на автоматические и автоматизированные устройства.

Использование ЭВМ приводит к коренной модернизации технологии производства практически во всех отраслях промышленности, коммерческой и финансово-кредитной деятельности, что является основой повышения производительности и улучшению условий труда людей. Именно поэтому современный выпускник вуза должен обладать общекультурными и профессиональными компетенциями в области информатики, программирования и практического использования средств вычислительной техники.

СПИСОК ЛИТЕРАТУРЫ

1. **Акулов, О. А.** Информатика: базовый курс : учеб. для студентов вузов, бакалавров магистров, обучающихся по направлениям 552800, 654600 «Информатика и вычислительная техника» / О. А. Акулов, Н. В. Медведев. – М. : Омега-Л, 2007. – 560 с.

2. **Острейковский, В. А.** Информатика : учеб. для вузов / В. А. Острейковский. – М. : Высш. шк., 1999. – 511 с.

3. **Горбатов, В. А.** Основы дискретной математики : учеб. пособие для студентов вузов / В. А. Горбатов. – М. : Высш. шк., 1986. – 311 с.

4. **Алферова, З. В.** Теория алгоритмов : учеб. пособие для студентов вузов / З. В. Алферова. – М. : Статистика, 1973. – 164 с.

5. **Вычислительная** техника и программирование : учеб. для студентов техн. вузов / А. В. Петров [и др.]. – М. : Высш. шк., 1990. – 479 с.

6. **Кнут, Д.** Искусство программирования для ЭВМ. Т. 1. Основные алгоритмы / Д. Кнут. – М. : Мир, 1976. – 736 с.

7. **Уинер, Р.** Язык Турбо Си / Р. Уинер. – М. : Мир, 1991. – 380 с.

8. **Фигурнов, В. Э.** IBM PC для пользователя. Изд. 5-е, исправл. и доп. / В. Э. Фигурнов. – СПб. : АО «Коруна», 1994. – 352 с.

9. **Смарт, Н.** Криптография / Н. Смарт. – М. : Техносфера, 2005. – 528 с.

10. **Основы криптографии :** учеб. пособие / А. П. Алферов [и др.]. – М. : Гелиос АРВ, 2002. – 480 с.

ОГЛАВЛЕНИЕ

Учебное издание

ИВАНОВА Ольга Геннадьевна,
КУЛАКОВ Юрий Владимирович,
ШАХОВ Николай Гурьевич,
ОДНОЛЬКО Валерий Григорьевич

ПРАКТИКУМ ПО ИНФОРМАТИКЕ

Учебное пособие

Редактор Л. В. Комбарова
Инженер по компьютерному макетированию И. В. Евсеева

Подписано в печать 06.11.2014.
Формат 60×84 /16. 6,51 усл. печ. л.
Тираж 100 экз. Заказ № 510

Издательско-полиграфический центр
ФГБОУ ВПО «ТГТУ»
392000, г. Тамбов, ул. Советская, д. 106, к. 14
Тел. 8(4752) 63-81-08;
E-mail: izdatelstvo@admin.tstu.ru