Министерство образования и науки Российской Федерации
Московский физико-технический институт
(государственный университет)
Заочная физико-техническая школа

ИНФОРМАТИКА и ИКТ

Математическая теория информации

Задание №2 для 11-х классов

(2014 – 2015 учебный год)

г. Долгопрудный, 2016

Составитель: В.В. Мерзляков, ассистент кафедры информатики СУНЦ МГУ.

Информатика: задание №2 для 11-х классов (2016 – 2017 учебный год), 2016, 20 с.

Дата отправления заданий по информатике - 10 декабря 2016 г.

Составитель:
Мерзляков Василий Владимирович

Подписано 07.11.16. Формат 60×90 1/16.
Бумага типографская. Печать офсетная. Усл. печ. л. 1,25
Уч.-изд. л. 1,11. Тираж 600. Заказ №44-з.

Заочная физико-техническая школа
Московского физико-технического института
(государственного университета)
ООО «Печатный салон ШАНС»

Институтский пер., 9, г. Долгопрудный, Москов. обл., 141700,
ЗФТШ, тел/факс (495) 408-51-45 – **заочное отделение,**
тел./факс (498)744-63-51 – **очно-заочное отделение,**
тел. (925) 755-55-80 – **очное отделение.**

e-mail: zftsh@mail.mipt.ru

Наш сайт: www.school.mipt.ru

§1. Основы теории информации

1.1. Понятие информации. Количество информации. Единицы измерения информации

Информация является одним из фундаментальных понятий современной науки наряду с такими понятиями, как «вещество» и «энергия».

Общее определение этому термину дать невозможно. Однако в различных предметных областях даётся специализированное определение информации, подходящее для данной предметной области. В рамках этого задания мы будем говорить о математической теории информации и рассмотрим два подхода – содержательный (Клод Шеннон) и алфавитный (А.Н.Колмогоров). Начнём с определения понятия «информация» в каждом из этих подходов.

Определение 1. В содержательном подходе, ***информация*** — это снятая *неопределённость*. ***Неопределённость*** некоторого события — это количество возможных результатов (исходов) данного события.

Например, если мы подбрасываем вверх монету, то она может упасть двумя различными способами (орлом вверх или решкой вверх). Соответственно, у данного события два возможных исхода. Если же подбрасывать игральный кубик, то исходов будет шесть.

Определение 2. В алфавитном подходе ***информация –*** это сообщение (последовательность символов некоторого алфавита). Причём существенными являются только размер алфавита и количество символов в сообщении. Конкретное содержание сообщения интереса не представляет. Чаще всего алфавит является двоичным (состоит из 2 символов – «0» и «1»).

После таких определений понятия «информация» можно говорить об её измерении. Введём несколько основных единиц измерения информации.

Чаще всего в качестве основной единицы измерения информации используется бит. При алфавитном подходе ***один бит*** — это количество информации, которое можно передать в сообщении, состоящем из одного двоичного знака («0» или «1»). С точки же зрения содержательного подхода ***один бит*** — это количество информации, уменьшающее неопределённость знания в два раза.

Наряду с битами можно использовать и другие единицы измерения информации, например, триты или диты. При алфавитном подходе ***один трит*** — это количество информации, которое можно передать в сообщении, состоящем из одного троичного знака («0», «1» или «2»). С точки же зрения содержательного подхода ***один трит*** — это количество информации, уменьшающее неопределённость знания в три раза.

Соответственно, ***один дит*** — это количество информации, уменьшающее неопределённость знания в десять раз, и количество информации, которое можно передать в сообщении, состоящем из одного десятичного знака (арабской цифры). В некоторых задачах (например, в задаче взлома кодового замка) удобнее в качестве основной единицы измерения информации использовать не биты, а диты, поскольку угадывание каждой цифры из кода уменьшает количество комбинаций в 10 раз.

Для каждой основной единицы измерения информации существуют производные более крупные единицы измерения. Поскольку чаще всего мы будем использовать в качестве основной единицы бит, рассмотрим производные единицы измерения для бита. На практике чаще всего используется не бит, а байт.

1 байт (1B) = 8 бит;

Далее существует две линейки производных единиц для байта – линейка ***десятичных приставок*** и линейка ***двоичных приставок***. В случае десятичных приставок каждая следующая единица измерения равна 1000 предыдущих единиц. Обозначаются десятичные приставки латинскими буквами (буква префикса из системы СИ и заглавная «B», обозначающая «байт») Итак:

1 килобайт (1 kB) = 1000 B (1000 байт);
1 мегабайт (1 MB) = 1000 kB;
1 гигабайт (1 GB) = 1000 MB;
1 терабайт (1 TB) = 1000 GB;
1 петабайт (1 PB) = 1000 TB;
1 эксабайт (1 EB) = 1000 PB;
1 зеттабайт (1 ZB) = 1000 EB;
1 йоттабайт(1 YB) = 1000 ZB.

Более крупных единиц на настоящий момент не введено.

При использовании двоичных приставок, каждая следующая единица измерения равна 1024 предыдущих единиц. В России принято обозначать двоичные приставки, записывая префикс заглавной русской буквой и после него слово «байт» целиком и тоже русскими буквами. За рубежом для обозначения двоичных приставок между префиксом и «B» добавляется маленькая буква «i» (от слова «binary»). Кроме того, все префиксы записываются заглавными буквами. Итак:

1 кибибайт (1 Кбайт, 1 KiB) = 2^{10} байт = 1024 байт;
1 мебибайт (1 Мбайт, 1 MiB) = 2^{20} байт = 1024 Кбайт;
1 гибибайт (1 Гбайт, 1 GiB) = 2^{30} байт = 1024 Мбайт;
1 тебибайт (1 Тбайт, 1 TiB) = 2^{40} байт = 1024 Гбайт;
1 пебибайт (1 Пбайт, 1 PiB) = 2^{50} байт = 1024 Тбайт;
1 эксбибайт (1 Эбайт, 1EiB) = 2^{60} байт = 1024 Пбайт;
1 зебибайт (1 Збайт, 1 ZiB) = 2^{70} байт = 1024 Эбайт;
1 йобибайт (1 Йбайт, 1 YiB) = 2^{80} байт = 1024 Збайт.

1.2. Формула Хартли измерения количества информации. Закон аддитивности информации

Как уже упоминалось выше, в качестве основной единицы измерения информации мы будем использовать бит. Соответственно, с точки зрения алфавитного подхода мы будем кодировать информацию при помощи нулей и единиц (двоичных знаков).

Определение. Для того чтобы измерить количество информации в сообщении, надо закодировать сообщение в виде последовательности нулей и единиц *наиболее рациональным способом*, позволяющим получить *самую короткую последовательность*. Длина полученной последовательности нулей и единиц и является **мерой количества информации** в битах.

Поставим себе одну из наиболее часто встречающихся задач в теории информации. Пусть у нас есть N возможных равновероятных вариантов исходов некоторого события. Какое количество информации нам нужно получить, чтобы оставить только один вариант?

Например, пусть мы знаем, что некоторая интересная для нас книга находится на одной из полок нашего книжного шкафа, в котором 8 полок. Какое количество информации нам нужно получить, чтобы однозначно узнать полку, на которой находится книга?

Решим эту задачу с точки зрения содержательного и алфавитного подходов. Поскольку изначально в шкафу было 8 полок, а в итоге мы выберем одну, следовательно, неопределённость знания о местоположении книги уменьшится в 8 раз. Мы говорили, что один бит – это количество информации, уменьшающее неопределённость знания в 2 раза. Следовательно, мы должны получить 3 бита информации.

Теперь попробуем использовать алфавитный подход. Закодируем номера всех полок при помощи 0 и 1. Получим следующие номера: 000, 001, 010, 011, 100, 101, 110, 111. Для того чтобы узнать, на какой полке находится книга, мы должны узнать номер этой полки. Каждый номер состоит из 3 двоичных знаков. А по определению, 1 бит (в алфавитном подходе) – это количество информации в сообщении, состоящем из 1 двоичного знака. То есть мы тоже получим 3 бита информации.

Прежде чем продолжить рассмотрение поставленной общей задачи введём важное математическое определение.

Определение. Назовём логарифмом числа N по основанию a такое число X**,** что $a^x = N$. Обозначение: $X = \log_a N$.

На параметры логарифма налагаются некоторые ограничения. Число N обязательно должно быть строго больше 0. Число a (основание логарифма) должно быть также строго больше нуля и при этом не равняться единице (ибо при возведении единицы в любую степень получается единица).

Теперь вернёмся к нашей задаче. Итак, какое же количество информации нам нужно получить, чтобы выбрать один исход из N равновероятных? Ответ на этот вопрос даёт формула Хартли: $H = \log_a N$, где N – это количество исходов, а H – количество информации, которое нужно получить для однозначного выбора 1 исхода. Основание логарифма обозначает единицу измерения количества информации. То есть если мы будем измерять количество информации в битах, то логарифм нужно брать по основанию 2, а если основной единицей измерения станет трит, то, соответственно, логарифм берётся по основанию 3.

Рассмотрим несколько примеров применения формулы Хартли.

Задача 1. *В библиотеке* 16 *стеллажей, в каждом стеллаже* 8 *полок. Какое количество информации несёт сообщение о том, что нужная книга находится на четвёртой полке?*

Решение. Решим эту задачу с точки зрения содержательного подхода. В переданном нам сообщении указан только номер полки, но не указан номер стеллажа. Таким образом, устранилась неопределённость, связанная с полкой, а стеллаж, на котором находится книга, мы всё ещё не знаем. Так как известно, что в каждом стеллаже по 8 полок, следовательно, неопределённость уменьшилась в 8 раз. Следовательно, количество информации можно вычислить по формуле Хартли $H=\log_2 8=3$ бита информации.

Задача 2. *Имеется* 27 *монет, одна из которых фальшивая и легче всех остальных. Сколько потребуется взвешиваний на двухчашечных весах, чтобы однозначно найти фальшивую монету?*

Решение. В этой задаче неудобно использовать бит в качестве основной единицы измерения информации. Двухчашечные весы могут принимать три положения: левая чаша перевесила, значит, фальшивая монета находится в правой; правая чаша перевесила, значит, монета находится в левой; или же весы оказались в равновесии, что означает отсутствие фальшивой монеты на весах. Таким образом, одно взвешивание может уменьшить неопределённость в три раза, следовательно, будем использовать в качестве основной единицы измерения количества информации трит.

По формуле Хартли $H = \log_3 27 = 3$ трита. Таким образом, мы видим, что для того чтобы найти фальшивую монету среди остальных, нам потребуется три взвешивания.

Логарифмы обладают очень важным свойством:

$$\log_a (X * Y) = \log_a X + \log_a Y.$$

Если переформулировать это свойство в терминах количества информации, то мы получим **закон аддитивности информации**: Количество информации $H(x_1, x_2)$, необходимое для установления пары (x_1, x_2), равно сумме количеств информации $H(x_1)$ и $H(x_2)$, необходимых для независимого установления элементов x_1 и x_2:

$$H(x_1, x_2) = H(x_1) + H(x_2).$$

Проиллюстрируем этот закон на примере. Пусть у нас есть игральная кость в форме октаэдра (с 8 гранями) и монета. И мы одновременно подбрасываем их вверх. Нужно узнать, какое количество информации несёт сообщение о верхней стороне монеты после падения (орёл или решка) и числе, выпавшему на игральной кости.

Игральная кость может упасть 8 различными способами, следовательно, по формуле Хартли можно вычислить, что, определив число, выпавшее на игральной кости, мы получаем 3 бита информации. Соответственно, монета может упасть только 2 способами и несёт в себе 1 бит информации. По закону аддитивности информации мы можем сложить полученные результаты и узнать, что интересующее нас сообщение несёт 4 бита информации.

Рассмотрим другой способ решения этой задачи. Если мы сразу рассмотрим все возможные исходы падения 2 предметов, то их будет 16 (кость выпадает 8 способами, а монета – орлом вверх, и кость выпадает 8 способами, а монета – решкой вверх). По формуле Хартли находим, что интересующее нас сообщение несёт 4 бита информации.

***Замечание*: если в результате вычислений по формуле Хартли получилось нецелое число, а в задаче требуется указать целое число бит, то результат следует округлить в большую сторону.**

1.3. Примеры решения задач по теме «Математическая теория информации»

Задача 1. *В велокроссе участвуют* 130 *спортсменов. Специальное устройство регистрирует прохождение каждым из участников промежуточного финиша, записывая его номер с использованием минимально возможного количества бит, одинакового для каждого спортсмена. Каков информационный объём сообщения, записанного устройством, после того как промежуточный финиш прошли* 75 *велосипедистов?*

Решение. Первым делом нужно определить, сколько бит необходимо для кодирования 130 номеров спортсменов. Поскольку номера записываются в некотором устройстве, количество бит для кодирования каждого номера обязательно должно быть целым: $H = \log_2 130$. После

округления результата в большую сторону получим число 8. Следовательно, для кодирования 1 номера необходим 1 байт. Таким образом, информационный объём сообщения, записанного устройством, составляет 75 байт.

Задача 2. *В некоторой стране автомобильный номер состоит из* 7 *символов. В качестве символов используют* 18 *различных букв и десятичные цифры в любом порядке.*

Каждый такой номер в компьютерной программе записывается минимально возможным и одинаковым целым количеством байтов, при этом используют посимвольное кодирование и все символы кодируются одинаковым и минимально возможным количеством битов.

Определите объём памяти, отводимый этой программой для записи 60 *номеров.*

Решение. Первое действие аналогично предыдущей задаче – нужно установить, каким количеством бит кодируется 1 символ. Всего используется 18 букв и 10 десятичных цифр, то есть 28 символов. По формуле Хартли $H = \log_2 28$. После округления получается 5 бит на 1 символ. Вторым действием нужно узнать, какой объём памяти занимает 1 номер. Поскольку номер состоит из 7 символов, а каждый символ кодируется 5 битами, нам потребуется 35 бит памяти для хранения 1 номера. Однако по условию каждый номер должен записываться целым количеством байтов, а в каждом байте 8 бит. Ближайшее сверху к 35 число, делящееся на 8 – это число 40, следовательно, на каждый номер отводится 5 байт. Таким образом, для записи 60 номеров программе потребуется 60*5 = 300 байт памяти.

Задача 3. *Сигналы с судна на берег передают, используя различное положение рук. Каждая рука может быть поднята вверх, отведена в сторону или опущена вниз. Сколько различных сигналов можно подать двумя руками, если важно то, какая рука была в каком положении, но обе руки могут находиться и в одинаковом положении?*

Решение. Главная ловушка этой задачи заключается в следующем неверном ходе мыслей: «Раз одной рукой передаётся 3 сигнала, значит, двумя в 2 раза больше, то есть 6». На самом деле число исходов с добавлением новой руки увеличивается в 3 раза, поскольку можно продублировать все положения первой руки для каждого из 3 возможных положений второй. Таким образом, в ответе получается 9 сигналов.

Задача 4. *В течение* 5 *секунд было передано сообщение, объём которого составил* 375 *байт. Каков размер алфавита, с помощью которого записано сообщение, если скорость его передачи составила* 200 *символов в секунду?*

Решение. Первым делом найдём скорость передачи этого сообщения: 375/5 = 75 байт в секунду. Далее, нам известно, что в секунду передавалось 200 символов, которые занимают 75 байт памяти. Поэтому следующим действием найдём объём памяти, отводимый под 1 символ, переведя ответ в биты (ибо уже из входных чисел очевидно, что под каждый символ отводится менее 1 байта): 75*8/200 = 600/200 = 3. Таким образом, под каждый символ отводится 3 бита.

Применяя формулу Хартли, находим, что алфавит состоит из 8 символов.

§ 2. Представление текстовой информации в компьютере

Всякий текст состоит из ***символов*** — букв, цифр, знаков препинания и т. д., — которые человек различает по начертанию. Однако для компьютерного представления текстовой информации такой метод неудобен, а для компьютерной обработки текстов — и вовсе неприемлем. Используется другой способ: все символы кодируются числами, и текст представляется в виде набора чисел — ***кодов символов,*** его составляющих. При выводе текста на экран монитора или принтер необходимо восстановить изображения всех символов, составляющих данный текст. Для этого используются ***кодовые таблицы символов***, в которых для каждого символа устанавливается соответствие между его кодом и изображением. Все кодовые таблицы, используемые в любых компьютерах и любых операционных системах, подчиняются международным стандартам кодирования символов.

Основой для компьютерных стандартов кодирования символов послужил ASCII (***A****merican* ***S****tandard* ***C****ode for* ***I****nformation* ***I****nterchange*) — американский стандартный код для обмена информацией, разработанный в 1960-х годах и применяемый в США для любых видов передачи информации. В нём используется 7-битное кодирование: общее количество символов составляет 2^7=128, из них первые 32 символа — «управляющие», а остальные — «изображаемые», т. е. имеющие графическое изображение. Управляющие символы должны восприниматься устройством вывода текста как команды, например:

Символ	Действие	Английское название
№7	Подача стандартного звукового сигнала	Beep
№8	Затереть предыдущий символ	Back Space (BS)
№13	Перевод строки	Line Feed (LF)
№26	Конец текстового файла	End Of File (EOF)
№27	Отмена предыдущего ввода	Escape (ESC)

К изображаемым символам в ASCII относятся буквы английского (латинского) алфавита (заглавные и прописные), цифры, знаки препинания и арифметических операций, скобки и некоторые специальные символы. Фрагмент кодировки ASCII приведён в таблице.

Символ	Десятичный код	Двоичный код	Символ	Десятичный код	Двоичный код
Пробел	32	00100000	0	48	00110000
!	33	00100001	1	49	00110001
#	35	00100011	2	50	00110010
$	36	00100100	3	51	00110011
*	42	00101010	4	52	00110100
+	43	00101011	5	53	00110101
,	44	00101100	6	54	00110110
–	45	00101101	7	55	00110111
.	46	00101110	8	56	00111000
/	47	00101111	9	57	00111001
A	65	01000001	N	78	01001110
B	66	01000010	O	79	01001111
C	67	01000011	P	80	01010000
D	68	01000100	Q	81	01010001
E	69	01000101	R	82	01010010
F	70	01000110	S	83	01010011
G	71	01000111	T	84	01010100
H	72	01001000	U	85	01010101
I	73	01001001	V	86	01010110
J	74	01001010	W	87	01010111
K	75	01001011	X	88	01011000
L	76	01001100	Y	89	01011001
M	77	01001101	Z	90	01011010

Хотя в ASCII символы кодируются 7-ю битами, в памяти компьютера под каждый символ отводится ровно 1 байт (8 бит). И получается, что один бит из каждого байта не используется.

Главный недостаток стандарта ASCII заключается в том, что он рассчитан на передачу только текста, состоящего из английских букв. Со временем возникла необходимость кодирования и неанглийских букв. Во многих странах для этого стали разрабатывать *расширения ASCII-кодировки*, в которых применялись однобайтные коды символов; при этом первые 128 символов кодовой таблицы совпадали с кодировкой ASCII, а остальные (со 128-го по 255-й) использовались для кодирования букв национального алфавита, символов национальной валюты и т. п. Из-за несогласованности этих разработок для многих языков было создано по нескольку вариантов кодовых таблиц (например, для русского языка их около десятка).

Впоследствии использование кодовых таблиц было несколько упорядочено: каждой кодовой таблице было присвоено особое название и номер. Указав кодовую таблицу, автоматически выбирают и язык, которым можно пользоваться в дополнение к английскому; точнее, выбирается то, как будут интерпретироваться символы с кодами более 127.

Для русского языка наиболее распространёнными являются однобайтовые кодовые таблицы CP-866, Windows-1251, ISO 8859-5 и КОИ-8. В них первые 128 символов совпадают с ASCII-кодировкой, а русские буквы помещены во второй части таблицы (с номерами 128-255), однако коды русских букв в этих кодировках различны! Сравните, например, кодировки КОИ-8 (Код Обмена Информацией 8-битный, международное название «koi-8r») и Windows-1251, фрагменты которых приведены в таблицах на странице 13.

Несовпадение кодовых таблиц приводит к ряду неприятных эффектов: один и тот же текст (неанглийский) имеет различное компьютерное представление в разных кодировках, соответственно, текст, набранный в одной кодировке, будет нечитабельным в другой!

Однобайтовые кодировки обладают одним серьёзным ограничением: количество различных кодов символов в отдельно взятой кодировке недостаточно велико, чтобы можно было пользоваться одновременно несколькими языками. Для устранения этого ограничения в 1993-м году был разработан новый стандарт кодирования символов, получивший название Unicode, который, по замыслу его разработчиков, позволил бы использовать в текстах любые символы всех языков мира.

Кодировка KOI-8

─ 128	│ 129	┌ 130	┐ 131	└ 132	┘ 133	├ 134	┤ 135	┬ 136	┴ 137	┼ 138	▀ 139	▄ 140	█ 141	▌ 142	▐ 143
░ 144	▒ 145	▓ 146	⌠ 147	■ 148	∙ 149	√ 150	≈ 151	≤ 152	≥ 153	nbsp 154	⌡ 155	° 156	² 157	· 158	÷ 159
═ 160	║ 161	╒ 162	ё 163	╓ 164	╔ 165	╕ 166	╖ 167	╗ 168	╘ 169	╙ 170	╚ 171	╛ 172	╜ 173	╝ 174	╞ 175
╟ 176	╠ 177	╡ 178	Ё 179	╢ 180	╣ 181	╤ 182	╥ 183	╦ 184	╧ 185	╨ 186	╩ 187	╪ 188	╫ 189	╬ 190	© 191
ю 192	а 193	б 194	ц 195	д 196	е 197	ф 198	г 199	х 200	и 201	й 202	к 203	л 204	м 205	н 206	о 207
п 208	я 209	р 210	с 211	т 212	у 213	ж 214	в 215	ь 216	ы 217	з 218	ш 219	э 220	щ 221	ч 222	ъ 223
Ю 224	А 225	Б 226	Ц 227	Д 228	Е 229	Ф 230	Г 231	Х 232	И 233	Й 234	К 235	Л 236	М 237	Н 238	О 239
П 240	Я 241	Р 242	С 243	Т 244	У 245	Ж 246	В 247	Ь 248	Ы 249	З 250	Ш 251	Э 252	Щ 253	Ч 254	Ъ 255

Кодировка Windows 1251

Ѓ 128	ѓ 129	‚ 130	є 131	„ 132	… 133	† 134	‡ 135	€ 136	‰ 137	Ѕ 138	‹ 139	њ 140	Ќ 141	ћ 142	Ў 143
ў 144	‘ 145	’ 146	“ 147	” 148	• 149	– 150	— 151	ё 152	™ 153	ѕ 154	› 155	ќ 156	и 157	ґ 158	ї 159
nbsp 160	Ў 161	ў 162	Ј 163	¤ 164	Ґ 165	¦ 166	§ 167	Ё 168	© 169	Є 170	« 171	¬ 172	shy 173	® 174	Ї 175
° 176	± 177	І 178	і 179	ґ 180	µ 181	¶ 182	· 183	ё 184	№ 185	є 186	» 187	ј 188	Ѕ 189	ѕ 190	ї 191
А 192	Б 193	В 194	Г 195	Д 196	Е 197	Ж 198	З 199	И 200	Й 201	К 202	Л 203	М 204	Н 205	О 206	П 207
Р 208	С 209	Т 210	У 211	Ф 212	Х 213	Ц 214	Ч 215	Ш 216	Щ 217	Ъ 218	Ы 219	Ь 220	Э 221	Ю 222	Я 223
а 224	б 225	в 226	г 227	д 228	е 229	ж 230	з 231	и 232	й 233	к 234	л 235	м 236	н 237	о 238	п 239
р 240	с 241	т 242	у 243	ф 244	х 245	ц 246	ч 247	ш 248	щ 249	ъ 250	ы 251	ь 252	э 253	ю 254	я 255

В Unicode на кодирование символов отводится 32 бита. Первые 128 символов (коды 0-127) совпадают с таблицей ASCII, все основные алфавиты современных языков полностью умещаются в первые 65536 кодов ($65536=2^{16}$), а в целом стандарт Unicode описывает все алфавиты современных и мёртвых языков; для языков, имеющих несколько алфавитов или вариантов написания (например, японский и индий-

ский), закодированы все варианты; внесены все математические и иные научные символьные обозначения, и даже — некоторые придуманные языки (например, письменности эльфов и Мордора из эпических произведений Дж.Р.Р. Толкиена). Потенциальная информационная ёмкость Unicode столь велика, что сейчас используется менее одной тысячной части возможных кодов символов!

В современных компьютерах и операционных системах используется укороченная, 16-битная версия Unicode, в которую входят все современные алфавиты; эта часть Unicode называется базовой многоязыковой страницей (Base Multilingual Plane, BMP).

§3. Символьный тип данных в языке Паскаль

Теперь применим полученные знания о представлении текстовой информации на практике. В языке программирования Паскаль для работы с текстовой информацией есть специальный символьный тип переменных, который называется char (от английского character). Переменные этого типа занимают в оперативной памяти по 1 байту и, соответственно, могут принимать 256 различных значений. Значениями переменных этого типа являются элементы какой-либо однобайтовой кодовой таблицы (например, KOI-8 или Windows-1251). Какие именно символы являются значениями данного типа, зависит от того, какая кодовая таблица используется в момент *выполнения* (а не написания) программы. То есть одна и та же программа, например, печатающая изображение всех символов кодовой таблицы, на компьютерах с различными текущими кодировками будет иметь различные результаты работы.

Переменным символьного типа можно присваивать значения при помощи оператора присваивания. При этом есть два способа записи символьных констант. Первый способ – записать явное изображение символа, заключив его в апострофы. Пусть, например, переменная C имеет тип char. Присвоим ей значение: C:= 'a'; Описанный способ записи символьных значений удобно применять практически всегда. Единственный недостаток этого способа заключается в том, что так невозможно представить служебные символы, которые не имеют явных изображений (в кодовой таблице это первые 32 символа). Поэтому существует ещё один способ записи символьных констант – сначала указать спецсимвол решётку (#), а потом код интересующего нас символа.

Например, C:=#13; Недостаток этого способа заключается в том, что нужно помнить коды всех символов, поэтому обычно его применяют только для записи символов без явного изображения.

Переменные типа char можно выводить на экран при помощи оператора вывода и вводить с клавиатуры. Апострофы при вводе набирать не нужно (каждый апостроф также будет считаться отдельным символом). Служебные символы вводятся следующим образом: нужно зажать alt и на правой цифровой клавиатуре набрать код символа (например, 13).

К переменным типа char можно применять операции сравнения (> , < , >= , <= , = , <>). При этом сравниваются коды символов и б*о*льшим признаётся символ, имеющий больший код (то есть символ, находящийся дальше от нулевого). Результатом операции сравнения является логическое значение – true или false.

Существует 5 стандартных функций для работы с переменными символьного типа:

Функция	**Действие**	**Тип аргумента**	**Тип результата**
`Ord(c)`	Выдаёт код символа	`Char`	`Integer`
`Chr(x)`	Выдаёт символ по коду	`Integer`	`Char`
`Succ(c)`	Выдаёт следующий символ кодовой таблицы. Не определена для последнего символа	`Char`	`Char`
`Pred(c)`	Выдаёт предыдущий символ кодовой таблицы. Не определена для нулевого символа	`Char`	`Char`
`Upcase(c)`	Если аргумент является строчной латинской буквой, превращает его в соответствующую заглавную. Иначе ничего не делает	`Char`	`Char`

Тип char является порядковым, то есть для каждого символа можно назвать его порядковый номер в типе, а также следующий и предшествующий элементы типа. Например, символ '1' имеет код 49, следующий символ – это '2', а предыдущий – '0'. Благодаря этому свойству переменные типа char могут использоваться в качестве счётчиков в цикле for. Например, распечатать все заглавные латинские буквы можно следующим образом:

```
For  c:= 'A' to 'Z' do write (c);
```

где переменная с имеет тип char.

Если в цикле **for** используется слово **to**, то на каждом шаге цикла счётчик будет принимать следующее значение в типе, в случае же **downto** – предыдущее значение в типе.

Рассмотрим несколько примеров задач на символьные переменные.

Задача 1. *Вывести на экран все символы кодовой таблицы.*

Решение. Эту задачу можно решать двумя способами: перебрать все символы или все их коды – разница только в типе счётчика цикла.

Способ 1:

```
var c:char;
 begin
    for c:=#0 to #255 do
       write(ord(c),'-',c,' ');
     readln
end.
```

Способ 2:

```
var i:integer;
begin
   for i:=0 to 255 do
      write(i, '-',chr(i), ' ');
   readln
end.
```

Задача 2. *Дана последовательность символов, заканчивающаяся точкой. Подсчитайте сумму цифр, входящих в эту последовательность.*

Решение. Эта задача демонстрирует очень важную вещь – как превратить символ-цифру в целое число. Это осуществляется следующим образом: необходимо вычислить код интересующего нас символа (например, код единицы 49) и вычесть из него код символа «ноль». В любой кодировочной таблице символы-цифры идут подряд, поэтому, выполнив указанные действия, мы гарантированно получим числовое значение символа-цифры. Приведём полный текст решения.

```
var c: char; s: integer;
begin
   s :=0;
   read (c);
   while c <> '.' do
     begin
     if (c >= '0')and(c <= '9')
       then s:= s+ord(c)-ord('0');
         read (c);
       end;
   writeln ('s=',s);
   readln
  end.
```

Задача 3. *Дана непустая последовательность слов, состоящих из заглавных и строчных латинских букв в любом порядке. Между соседними словами запятая, за последним словом – точка. Никакие другие символы в последовательность не входят. Определить количество слов, которые начинаются на букву Z.*

Решение. Это ещё один классический тип задач на обработку последовательностей символов. При её решении у нас возникнет конструкция из вложенных циклов: внутренний цикл анализирует слово, а внешний перебирает слова. Приведём полный текст решения.

```
var c:char; s:integer;
begin
   s:=0;
   repeat
     read(c);
     if c='Z' then s:=s+1;
     repeat
       read(c)
     until (c=',')or(c='.')
   until c='.';
   writeln('s=',s);
   readln
end.
```

§4. Оператор выбора Case

Данный оператор представляет собой естественное расширение условного оператора. В общем виде он записывается следующим образом:

```
case <выражение порядкового типа> of
  константа_1: оператор_1;
  константа_2: оператор_2;
        ...
  Константа_n: оператор_n;
  else оператор
end
```

Слова: `case, of, else, end` - являются ключевыми словами языка. Выражение, стоящее между словами `case` и `of`, называется **селектором** и должно иметь порядковый тип. Тип является порядковым, если можно для каждого значения назвать порядковый номер в типе, предыдущее и следующее значение в типе (кроме первого и последнего значения в типе). Из известных нам стандартных типов порядковыми являются типы `integer, longint, boolean` и `char`. Тип `real` порядковым не является.

Работает оператор выбора следующим образом. Сначала вычисляется значение селектора, затем оно сравнивается с константами. В случае совпадения селектора с какой-нибудь константой выполняется оператор, стоящий после этой константы, далее управление переходит на следующий за `case` оператор программы. Если селектор не совпал ни с одной из констант, то выполняется оператор после слова else. Очевидно, что селектор и константы должны иметь одинаковые типы. Иначе невозможно будет провести операции сравнения.

Если нужно для многих различных значений селектора выполнить один и тот же набор команд, то можно не записывать множество строк с одинаковой правой частью, а перечислить константы через запятую, затем поставить двоеточие и один раз написать нужную последовательность команд. Если константы идут подряд, можно также записать их в виде диапазона: `константа_1..константа_2`. В этом случае команда будет выполняться при совпадении селектора с любой константой из диапазона. Граничные значения считаются включёнными в диапазон. Можно также указать несколько диапазонов через запятую.

Оператор выбора предполагает однозначный выбор варианта. То есть нельзя одной и той же константой пометить два различных варианта. Все константы должны быть различны. Особенно аккуратно следует обращаться с диапазонами. Широко распространённая ошибка – указывать одну и ту же константу в качестве начальной границы одно-

го диапазона и конечной – другого. Однако поскольку границы входят в диапазон, получается, что это значение будет входить в два разных диапазона.

Последнее замечание заключается в том, что в отличие от оператора if перед else необходимо ставить точку с запятой. И кстати, аналогично оператору if, если в ветке else должен стоять пустой оператор, её можно не записывать. Приведём примеры нескольких различных операторов варианта.

Пример 1.

```
case c of
'+': x := x + y;
'-': x := x - y;
'*': x := x * y;
else writeln('error')
end;
```

Пример 2.

```
case c of
'a'..'z','A'..'Z': writeln('letter');
'0'..'9':          writeln('digit')
end;
```

Контрольные вопросы

1(1). При угадывании целого числа в диапазоне от 1 до N было получено чуть менее 4 бит информации. Чему может быть равно число N?

2(1). Информационная ёмкость человеческой яйцеклетки приблизительно равна 2^{31} бит. Уместится ли информация о ней на одном носителе ёмкостью 0,8 Гбайт? Ответ обосновать.

3(1). Сколько полных Эксбибайт содержится в 2^{65} Кибибайтах?

4.(1). Можно ли за четыре взвешивания на двухчашечных весах найти фальшивую монету среди 70 монет, если известно, что она легче всех остальных? Ответ обосновать.

5.(2). Сколько можно построить различных последовательностей, состоящих не менее чем из 2 и не более чем из 5 четверичных знаков? (Например, из 0, 1, 2, 3)

6(2). Сообщение записывается с помощью алфавита, состоящего из 8 символов, которые кодируются так, чтобы в каждом байте было записано целое число символов. За секунду передается 1024 бит. Какова скорость передачи сообщения в символах в секунду? Ответ обосновать.

7(1). Сколько памяти отводится под переменные типа char? Какие значения могут принимать переменные этого типа?

8(1). Опишите два способа записи символьных констант в тексте программы. Укажите их достоинства и недостатки.

9(3). Вычислить значение выражения или указать на ошибочность записи.

a) succ(succ('0'))=pred('3') b) ord('R')–ord('9') c) 'm'*'4'
d) ord('d')+2<ord('g')
e) 'R'>'q' f) ord('w')<100 g) chr(35)<chr(66)
h) ord(succ(chr(44)))
i) chr(pred(ord('5')/5)+5) j) pred(chr(0))

Задачи

Внимание! В задачах 6, 7, 8 перед написанием собственно текста программы необходимо объяснить свой алгоритм на русском языке и прокомментировать все используемые переменные. Программы без пояснений проверяться не будут!

1(2). У Сарумана сломался палантир, и теперь ему придётся использовать систему сигналов для того, чтобы контролировать свою армию. Для контроля необходимо использовать 800 различных сигналов! Саруман может передавать сигналы, стоя на крыше своей башни и меняя положения рук. Каждую руку он может поднять вверх, отвести в сторону или опустить вниз (3 положения каждой руки). При этом, если обе руки не находятся в одинаковом положении, то на ладони каждой руки он может зажечь магический огонь одного из шести цветов. Если же руки находятся в одинаковом положении, то на ладони каждой можно зажечь магический огонь одного из 14 цветов. Без магического огня на обеих ладонях сигнал не подаётся. Сможет ли Саруман контролировать свою армию? Ответ обосновать.

2(2). Для хранения произвольного растрового изображения размером 256×1024 пикселей отведено 256 Кбайт памяти, при этом для каждого пикселя хранится двоичное число – код цвета этого пикселя. Для каждого пикселя для хранения кода выделено одинаковое количество бит. Сжатие данных не производится. Какое максимальное количество цветов можно использовать в изображении?

3(2). Сколько секунд потребуется модему, передающему сообщения со скоростью 48000 бит/с, чтобы передать цветное растровое изображение размером 1200×600 пикселей, при условии, что цветовая палитра изображения состоит из 65536 цветов?

4(2). В некоторой стране автомобильный номер состоит из одной буквы и трёх десятичных цифр, записанных после данной буквы. Среди цифр может быть не более одной семёрки и не более двух нулей. Сколько различных номеров можно построить таким образом?

5(2). Автомобильный номер в другой стране состоит из 7 символов. В качестве символов используют 20 различных букв и десятичные цифры в любом порядке. При этом каждая буква может использоваться в двух начертаниях – строчном и заглавном. Каждый такой номер в компьютерной программе записывается минимально возможным и одинаковым целым количеством байтов, при этом используют посимвольное кодирование и все символы кодируются одинаковым и минимально возможным количеством битов. Определите объём памяти, отводимый этой программой для записи 100 номеров.

6(4). На вход программе подаётся последовательность символов, заканчивающаяся точкой (точка – признак конца и в последовательность не входит). Вывести слово YES, если введённая последовательность является правильной записью нечётного числа в семнадцатеричной системе счисления, и NO в противном случае.

7(5). На вход программе подаётся последовательность символов, заканчивающаяся точкой (точка – признак конца и в последовательность не входит). Проверить, правильно ли в данной последовательности расставлены круглые скобки (то есть, справа от каждой открывающей есть соответствующая закрывающая, а слева от каждой закрывающей есть соответствующая открывающая). Например, в последовательности ()()(()) скобки расставлены правильно, а в последовательности ()) – неправильно. Вывести на экран в качестве ответа YES или NO.

8(6). На вход программе подаётся непустая последовательность из слов, состоящих из малых латинских букв. Между соседними словами – запятая, за последним словом – точка. Подсчитать количество слов, буквы в которых упорядочены в алфавитном порядке (одинаковые буквы могут идти подряд). При этом, в некоторых словах могут встречаться «ошибочные символы» (всё, что угодно, кроме малых латинских букв). Такие слова учитывать не нужно, даже если буквы в них стоят в алфавитном порядке. Сложные типы данных не использовать.

9(8). На вход программе подаётся последовательность символов, заканчивающаяся точкой (точка – признак конца и в последовательность не входит). Данная последовательность является правильной записью арифметического выражения, состоящего из целых десятичных чисел и знаков «плюс» и «минус» (могут стоять и перед первым числом). При этом два знака подряд в выражении не допускается. Программа должна вычислить значения данного выражения и вывести его на экран. Гарантируется, что ни на каком этапе вычислений тип longint не переполнится. Пример: введено «234-234+657.», программа должна вывести «657».