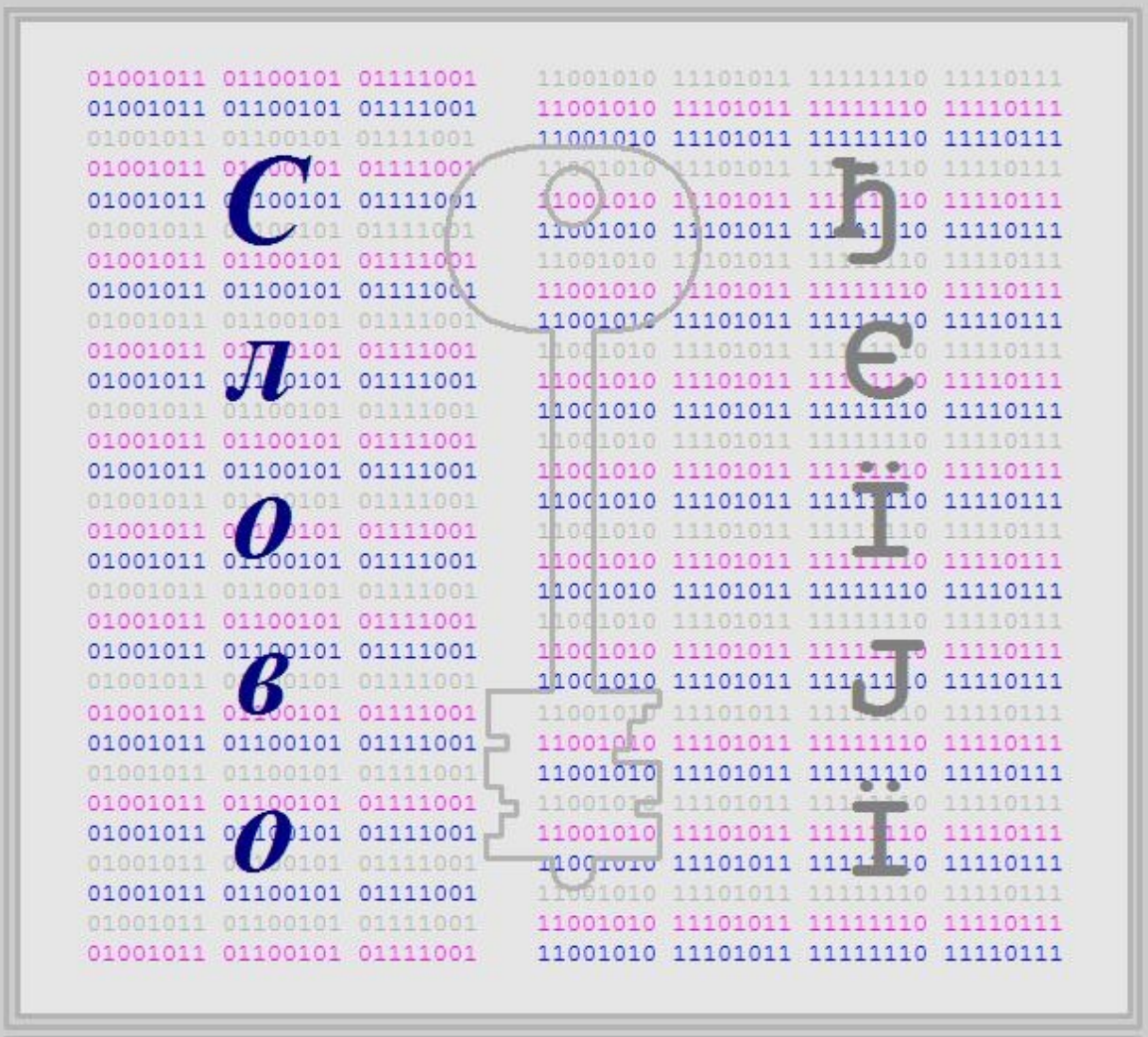


АЛЬТЕРНАТИВНАЯ ИНФОРМАТИКА

Маленький секрет в Большой сети

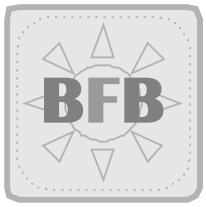
А.Н. Федяинов

BFB

АЛЬТЕРНАТИВНАЯ ИНФОРМАТИКА

А.Н. Федяинов

Маленький секрет в Большой сети



2010

Федяинов А.Н. Альтернативная информатика. Маленький секрет в Большой сети, 2010. 57 с.

Содержание брошюры построено на простом варианте преобразования читаемого текстового сообщения в кодированный текст.

В краткой форме рассмотрены: кодовая страница; запись числовых кодов знаков, символов в разных системах счета; операции с битами целочисленных типов данных. Изложение сопровождается программами на языке программирования Pascal.

Брошюра может рассматриваться как пособие по школьному курсу информатики.

Передать конструктивные предложения, замечания, направленные на улучшение содержания, можно:

электронной почтой

• bitword@mail.ru, beforbe@rambler.ru;

или обычной

• 398058 Липецк, 15 микрорайон, 10-33, Федяинову Александру Николаевичу.

СОДЕРЖАНИЕ

Маленький секрет в Большой сети

- Маленький секрет
- Есть задача – нужно знание основ предмета и общая схема решения
- Потребность, обдумывание и последовательные действия приводят к некоторым результатам

Как это работает: от текста к действиям с битами и обратно

- Текст, символы, множество символов, десятичные коды символов
- От десятичных чисел к двоичным числам (двоичной системе счета)
- Для перехода к десятичному представлению, число другой системы счета представляется полиномом
- Схема Горнера – алгоритм вычисления произвольного полинома
- Записываем числа в разных системах счета
- Двоичные числа хранятся, обрабатываются и передаются битами, байтами, машинными словами
- Поразрядные операции – действия с битами байтов, двоичными числами
- Простое поразрядное шифрование
- Возвращаясь к тексту, что далее
- Используемый модуль XORCrypt.pas
- Используемый модуль Mix.pas

Copyright – дословно, право делать копии

Маленький секрет в Большой сети

Маленький секрет

Уже в древние времена у человека была потребность сделать текст непонятным для непосвященных лиц, существовала тайнопись.

У народов Европы еще в первые века нашей эры появилась руническая письменность – руны или рунические надписи. Руны состоят из вертикальных и косых прямых линий, и не известно, на основе какого алфавита возникла руническая письменность. Руны обозначали не только звуки, но и целые понятия.

Руны наносились на предметы, и их магия использовалась для лечения болезней, побед в битвах, завоевания любви. Иногда руны были специально зашифрованы – существовала руническая тайнопись. В надписях слова могли разделяться точками, слово могло быть разделено на части или слова могли быть написаны слитно.

В последующие века тайнопись, способы тайной переписки развиваются. Например, для тайнописи используются симпатические чернила – бесцветная или слабоокрашенная жидкость. Сообщение, записанное симпатическими чернилами, невидимо в обычных условиях и выявляется специальными способами.

Элементы тайнописи существуют даже в литературе, в искусстве.

В литературе – эзопов язык, или язык иносказаний, маскирующий мысль автора.

Пример из живописи – картины Пикассо. Французский живописец 20 века Пабло Пикассо, оказавший огромное влияние на художников всех стран, в композициях нередко использовал тайнопись – зашифрованные номера телефонов и домов, части имен, названий.

Конечно, и в наши дни в жизни каждого человека есть хотя бы небольшие тайны, маленькие секреты, которые оказывают влияние на его взаимоотношения с окружающими людьми, на его признание окружающими людьми, и, в конечном счете, на его жизнь.

Некоторые такие небольшие тайны лучше держать в секрете. В определенных случаях при передаче важной информации, не будет лишним

хотя бы немного подстраховаться, закрыть свою информацию от тех, для кого она не предназначена.

С этим непосредственно связано понятие тайны переписки. Тайна переписки – это одна из конституционных свобод гражданина, она означает неприкосновенность почтовой и другой корреспонденции; эта тайна включает в себя также и неприкосновенность разговоров по телефону, сообщений по телеграфу.

Для дальнейшего изложения можно было бы придумать псевдо-литературный сюжет и ввести персонажи.

Например, Саня и Маша; они ведут переписку, пользуясь некоторыми возможностями Большой сети, и иногда оставляют свои небольшие сообщения на Общем ресурсе. Время от времени у них появляются маленькие секреты, которые они хотели бы сохранить в тайне.

Саня решил, что ему нужен простой способ или алгоритм изменения визуального представления сообщений, который позволит ему писать на "языке", известном лишь адресату – получателю, которому адресовано сообщение. Для этого Саня решил попытаться зашифровать, или хотя бы перевести в код некоторые свои сообщения.

Обстоятельства сложились в пользу Сани, так как у него были:

• личная заинтересованность добиться цели, представляющей интерес для многих людей;

• готовность думать и лично серьезно потрудиться для достижения цели;

• способность пойти на определенные жертвы – времени, усилий и средств для решения задачи.

И через некоторое время у него кое-что начинает получаться.

А что и как получается . . . об этом далее.

Впрочем, так как это изложение не является сказочно-повествовательным произведением, то сюжет, а точнее контент, будет другим.

Есть задача – нужно знание основ предмета и общая схема решения

Для доверительной, секретной переписки, передачи текста секретных документов по открытым, незащищенным каналам связи применяется шифрование.

Шифрование – преобразование читаемого сообщения или документа в нечитаемое; расшифровывание – обратная процедура (получение читаемого сообщения или документа из нечитаемого).

В общем случае, шифрование применяется для защиты от несанкционированного доступа к информации, содержащейся в данных, сообщениях, документах, т.е. для преобразования отдельных элементов, частей или всей информации в такой вид, чтобы существенно терялась возможность чтения первоначальной информации.

Шифр (англ. cipher) – метод, процедура или алгоритм секретного письма, использующие

• замену (подстановку) букв, символов и/или

• перенос (перестановку) обычного порядка букв (например, в алфавите), знаков в кодовой странице

применительно к исходному сообщению или документу в соответствии с секретным ключом.

Алгоритм – точное предписание, которое задает вычислительный процесс, начинающийся с некоторых исходных данных и направленный на получение определенных исходными данными результата.

Алгоритмом является, например, возведение числа в некоторую степень; здесь исходными данными служат число (основание) и показатель степени, результат – произведение нескольких равных сомножителей (степень числа).

Понятие алгоритма занимает важное место в современной вычислительной математике. Умение решать задачу "в общем виде" означает владение некоторым алгоритмом. Другими словами, для решения конкретных однотипных задач принципиально необходимы общие для данного типа задач методы.

Общее понимание состоит в том, что исходными данными и результатами алгоритма могут служить самые разнообразные объекты. Можно говорить об алгоритме перевода с одного языка на другой, об алгоритмическом описании процессов управления простыми и сложными объектами. Поэтому понятие алгоритма является одним из основных в современных вычислительных системах.

Одно из значений слова ключ (англ. key) – средство доступа или управления. В шифровании, **ключ** – коды или параметры, используемые процедурой, алгоритмом шифрования и расшифровывания.

Распространены шифр замены или подстановочный шифр (substitutional cipher) и перестановочный шифр (transposition(al) cipher); возможна также их комбинация.

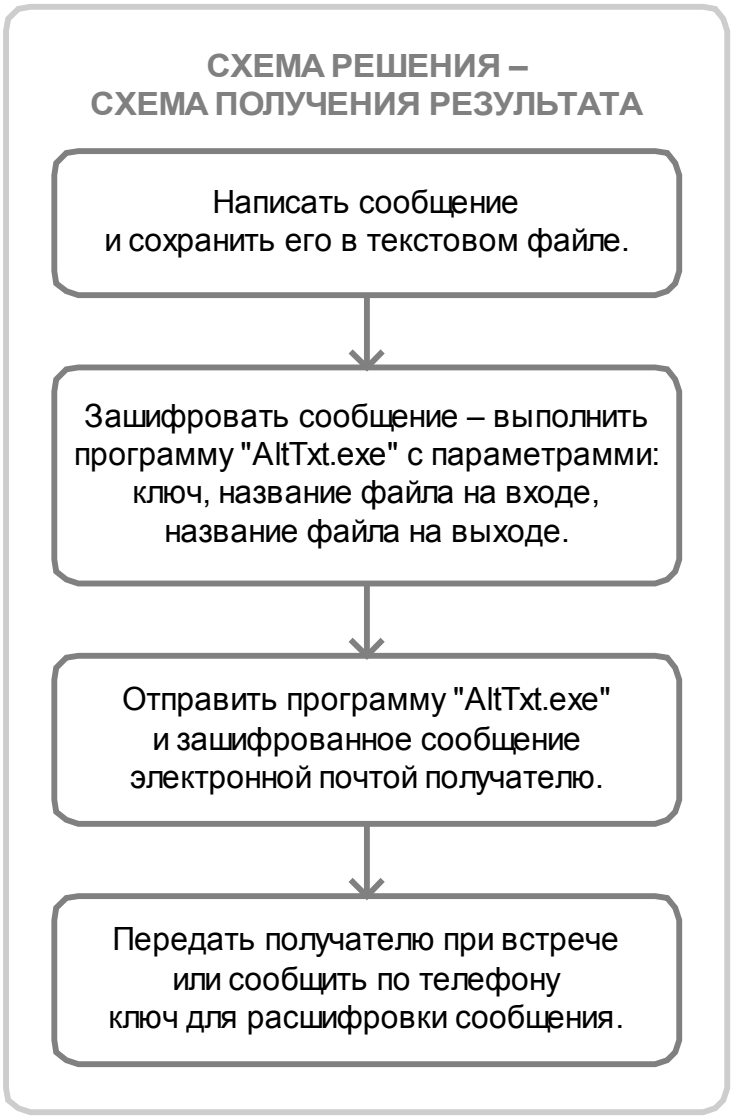
В простом случае, шифр замены – замещение одного знака (цифры, буквы, символа) другим, а шифр перестановки – циклическое смещение всех знаков

на определенное число позиций (например, букв в алфавите, чисел в системе чета, кодов в кодовой странице и т.д.).

Изучением шифров занимается криптография.

До начала решения задачи всегда полезно набросать (написать или нарисовать) общую схему получения желаемого результата.

В случае задачи простого шифрования (кодирования), такой схемой могла бы быть, например, следующая упорядоченная последовательность действий.



Потребность, обдумывание и последовательные действия приводят к некоторым результатам

Написанная на языке программирования Turbo Pascal, программа для простого шифрования (кодирования) текстового сообщения может быть, например, следующей.

```
program AltTxt;

uses
  XORCrypt,
  Mix;

begin
 {
   Function ParamStr(I: Word): String;
   returns the I-th parameter from the command line,
   or an empty string if Index is greater than ParamCount.
   ParamStr(0) returns the path and file name
   of the executing program.

   Function ParamCount: Word
   returns the number of parameters passed to the program
   on the command line.
 }

 {
   Example for the program parameters:
       alttxt.exe A textfile.txt crypto.txt
 }

  if (ParamCount <> 3) then
  begin
    Writeln('Three parameters are required',
      ' for this program:');
    Writeln('    AltTxt.exe Key Input Output');
    Writeln('Parameters:');
    Writeln('    Key    - secret key, any English letter;');
    Writeln('    Input  - input text file name',
      ' (e.g. intext.txt);');
    Writeln('    Output - output text file name',
      ' (e.g. outtext.txt).');
    WaitForExit('');
  end;

  ParStr := ParamStr(1);
  Ch := ParStr[1];
  CryptoKey := Ord(Ch);
```

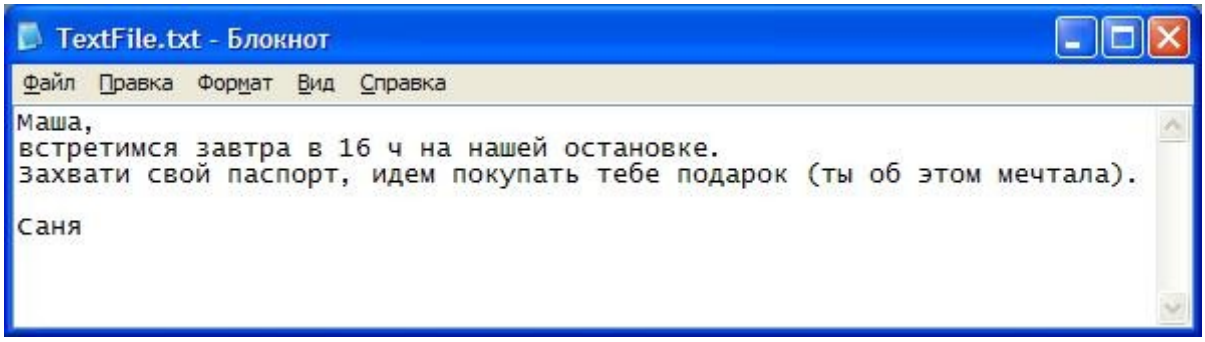
```
  InputFileName  := ParamStr(2);
  OutputFileName := ParamStr(3);

  XORTextCrypto(InputFileName, OutputFileName);
end.
```

Полный текст применяемых в этой программе процедур приведен далее под заголовками "Используемый модуль Mix.pas" и "Используемый модуль XORCrypt.pas".

Программа "alttxt.exe" работает следующим образом.

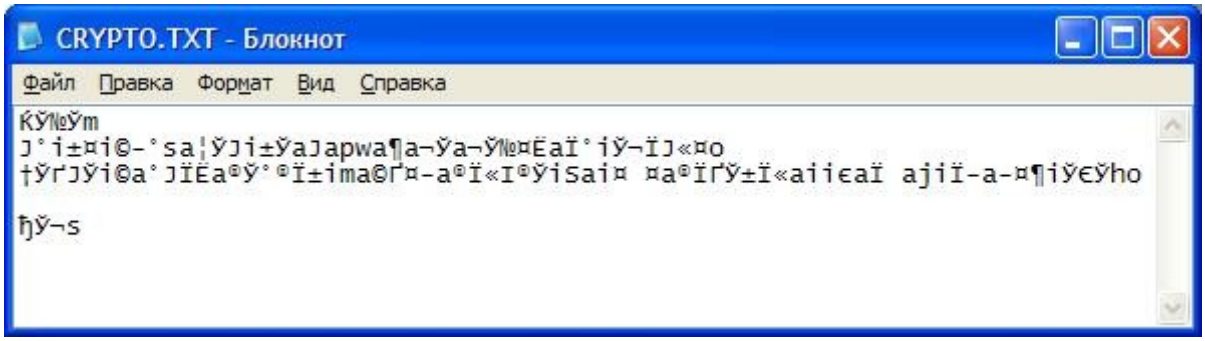
Наш Саня пишет сообщение для Маши:



Затем в операционной системе он выполняет команду:

```
alttxt.exe A textfile.txt crypto.txt
```

Для этого у него есть файл "Encode.bat"; результат работы программы – файл "CRYPTO.TXT":

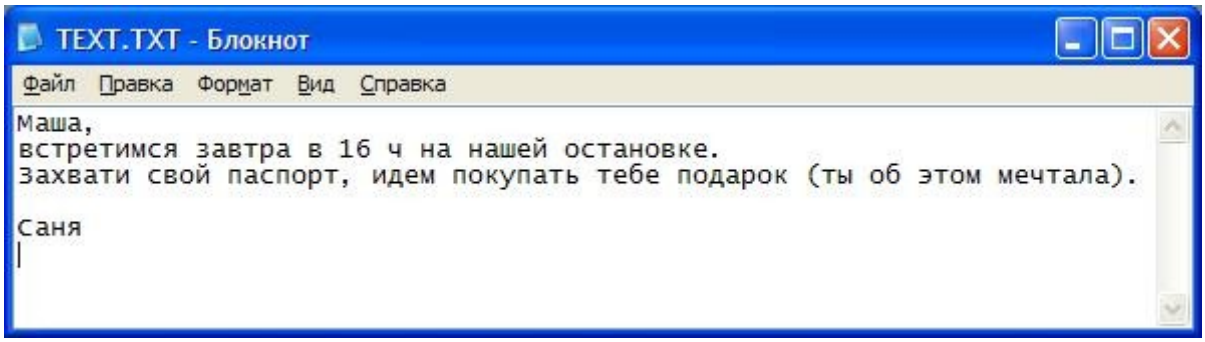


Саня оставляет этот файл на Общем ресурсе в Большой сети.

Маша, получив зашифрованное сообщение Сани и зная, что ключ в данное время – буква английского алфавита "A", выполняет в операционной системе команду:

```
alttxt.exe A crypto.txt text.txt
```

Маше для расшифровывания тоже удобнее применять командный файл, "Decode.bat"; в результате работы программы создается файл "TEXT.TXT" с расшифрованным сообщением:



Маша, конечно, рада, так как сообщение о важном предмете.

Как это работает: от текста к действиям с битами и обратно

Текст, символы, множество символов, десятичные коды символов

Текст (от лат. textus – ткань, соединение) – последовательность предложений, слов и знаков, построенная по правилам определенного языка и применяемой системы знаков (или символов).

Символ в вычислениях (character) – буква, цифра или другой знак, который применяется как составная часть организации, представления элементов данных или управления элементами данными.

Множество или **набор символов** (character set) содержит символы, необходимые определенному человеку для проведения значимого, поддающегося интерпретации взаимодействия с компьютером. В вычислениях символы подразделяются на управляющие (управляют информацией) и отображаемые или печатные (отображают данные, информацию, выводятся на печать).

Разные наборы символов содержат разные символы, алфавиты разных языков (латинский алфавит, русский и так далее) и различные группы условных графических знаков. Выделяются также группы знаков, обозначающие объекты, целые слова – идеографические группы (например, китайские или корейские иероглифы).

В вычислениях, **глиф** (glyph) – любой сгенерированный вычислительной системой знак, символ, рассматриваемый с точки зрения отображаемой формы и представления определенным сочетанием битов.

Код (code) – это совокупность символов (знаков) и система определенных правил, при помощи которых данные, информация могут быть представлены (закодированы) в виде набора из таких символов (знаков) для передачи, обработки и хранения.

Кодовая страница (code page, character encoding) представляет собой образование логических связей символов определенного набора и с их цифровым, байтовым представлением в вычислительных системах.

Какая-либо определенная кодовая страница часто изображается в форме таблицы, в которой символам данного набора соответствуют неотрицательные целые числа. Многие символы в разных кодовых страницах представлены разными числами. Важно заметить, что в вычислительной

системе все символы одной кодовой страницы представляются одинаковым количеством байтов.

Многие кодовые страницы, содержащие символы и буквы определенных языков, используют однобайтовое представление символов (содержат 256 символов), для других требуются дополнительные байты (как, например, в Unicode; Unicode содержит наборы знаков почти всех языков).

Кодовые страницы Windows – наборы символов или кодовые страницы, используемые в операционных системах Microsoft Windows, начиная с 1980-х и 1990-х годов. В качестве примера рассмотрим одну из таких кодовых страниц – с буквами русского алфавита.

Кодовая страница Windows 1251 (или кодовая страница CP1251) – имеет широко распространение и предназначена для охвата алфавитов с кириллической основой – русского, болгарского, сербского и других.

В кодовой странице CP1251 применяется 8-битовое кодирование символов (однобайтовое), и поэтому страница вмещает 256 символов. Для управляющих символов, как и во многих других кодовых страницах, использованы десятичные коды от 0 до 31 (всего 32 символа), а для отображаемых (печатных) символов – коды от 32 до 255 (всего 224 символа).

В языке программирования Pascal, символьный тип `Char` (character – знак, символ, буква) используется для представления знаков, символов, букв кодовой страницы. Символьные константы записываются между одиночными кавычками, например:

```
'D', '5', '+'.
```

Функция `Chr` преобразует целочисленное значение в соответствующий символ кодовой страницы.

Функция `Ord` возвращает для символа соответствующее целочисленное значение в кодовой странице.

Следующая программа записывает отображаемые (печатные) символы кодовой страницы CP1251.

Данные выводятся в текстовый файл "PRNCHARS.TXT" в форме, подготовленной для преобразования в таблицу; как разделитель выводимых значений использован (управляющий) символ табуляции. В программе на Turbo Pascal (Borland) управляющий символ записывается десятичным кодом символа со стоящим в начале знаком #; в данном случае – #9. Символы выводятся соответствующими им десятичными кодами в соответствии с кодовой страницей.

```
program PrnChars;

uses
{ List of unit dependencies goes here... }
  Mix;

begin
    CharacterTable('PrnChars.txt', 32, 255);
end.
```

Процедура `CharacterTable` полностью приведена далее, в тексте под заголовком "Используемый модуль Mix.pas".

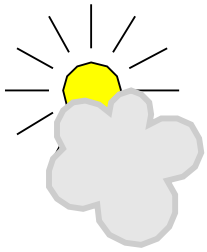
Результат выполнения программы перенесен в следующую таблицу.

(32)	! (33)	" (34)	# (35)	$ (36)	% (37)	& (38)	' (39)
((40)	) (41)	* (42)	+ (43)	, (44)	- (45)	. (46)	/ (47)
0 (48)	1 (49)	2 (50)	3 (51)	4 (52)	5 (53)	6 (54)	7 (55)
8 (56)	9 (57)	: (58)	; (59)	< (60)	= (61)	> (62)	? (63)
@ (64)	A (65)	B (66)	C (67)	D (68)	E (69)	F (70)	G (71)
H (72)	I (73)	J (74)	K (75)	L (76)	M (77)	N (78)	O (79)
P (80)	Q (81)	R (82)	S (83)	T (84)	U (85)	V (86)	W (87)
X (88)	Y (89)	Z (90)	[(91)	\ (92)	] (93)	^ (94)	_ (95)
` (96)	a (97)	b (98)	c (99)	d (100)	e (101)	f (102)	g (103)
h (104)	i (105)	j (106)	k (107)	l (108)	m (109)	n (110)	o (111)
p (112)	q (113)	r (114)	s (115)	t (116)	u (117)	v (118)	w (119)
x (120)	y (121)	z (122)	{ (123)	| (124)	} (125)	~ (126)	□ (127)
Ђ (128)	Ѓ (129)	, (130)	ѓ (131)	„ (132)	… (133)	† (134)	‡ (135)
€ (136)	‰ (137)	Љ (138)	‹ (139)	Њ (140)	Ќ (141)	Ћ (142)	Џ (143)
ђ (144)	' (145)	' (146)	" (147)	" (148)	• (149)	– (150)	— (151)
□ (152)	™ (153)	љ (154)	› (155)	њ (156)	ќ (157)	ћ (158)	џ (159)
(160)	Ў (161)	ў (162)	Ј (163)	¤ (164)	Ґ (165)	¦ (166)	§ (167)
Ё (168)	© (169)	Є (170)	« (171)	¬ (172)	(173)	® (174)	Ї (175)
° (176)	± (177)	І (178)	і (179)	ґ (180)	µ (181)	¶ (182)	· (183)
ё (184)	№ (185)	є (186)	» (187)	ј (188)	S (189)	s (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)

Задача упрощается!

Эта замечательная таблица показывает, что для решения нашей задачи из бесконечного множества чисел нас интересуют в первую очередь целые и неотрицательные числа, в основном, в интервале от 0 до 255.



От десятичных чисел к двоичным числам (двоичной системе счета)

Система счета, счисления (или нумерация) – способ выражения, обозначения чисел.

Цифры (от лат. cifra) – знаки (или символы) для обозначения чисел.

Основание системы счета (base) – это число различных цифр в системе счета, включая нуль. Роль основания может играть любое целое число, большее единицы.

Тогда, если основание какой-либо системы счета равно b, то цифры – это одиночные знаки для представления целых чисел от 0 до b-1 в этой системе.

Для основания какой-либо системы счета выполняется условие: b>1. Слишком маленькое основание может вызвать некоторое неудобство записи сравнительно больших чисел (строка цифр получается длиннее); с другой стороны, слишком большое основание потребовало бы заучивания многих цифр (знаков) и знания расширенной таблицы умножения.

В десятичной системе счета, как известно, цифрами называются знаки

0, 1, 2, 3, 4, 5, 6, 7, 8, 9

для обозначения нуля и девяти целых чисел, а основание системы равно 10.

Рассмотрим схематично, как записываются целые числа в десятичной системе. Начиная с нуля, будем последовательно увеличивать число на

единицу: 0, 1, 2, . . . , 9. Когда число стало равным 9, прибавив 1 видим, что только разрядом единиц это число не выражается – нет цифры (одноразрядного числа) для обозначения десяти. Тогда делаем перенос в следующий разряд – получим единицу следующего разряда (десятков), а число разряда единиц обнуляем; для обозначения результата слева от разряда единиц пишем 1, а в разряде единиц – 0, число десять выразится двумя разрядами как 10. Продолжая последовательно увеличивать число на единицу, получим разряды сотен, тысяч и т.д.

Тогда, например, запись десятичного числа 5474 обозначает:

$$5474 = 5\times1000 + 4\times100 + 7\times10 + 4$$
$$= 5\times10^3 + 4\times10^2 + 7\times10^1 + 4\times10^0.$$

Так как любое число в степени 0 равно 1, то и $10^0 = 1$.

Запись многозначного числа – это последовательность однозначных целых остатков. Иначе говоря, цифры в записи десятичного числа – это целые остатки, полученные при целочисленном делении числа на последовательные степени основания системы:

	5474	10			
Остаток – число единиц, равно	**4**	547	10		
Остаток – число десятков, равно		**7**	54	10	
Остаток – число сотен, равно			**4**	5	10
Остаток – число тысяч, равно				**5**	0

Разряд в арифметике – место, занимаемое цифрой при обозначении числа и соответствующее этому месту значение в системе счета.

В арифметике, в десятичной записи числа цифры 1-го разряда – единицы ($10^0 = 1$), 2-го – десятки (10^1) , 3-го – сотни (10^2) , 4-го – тысячи (10^3) и т. д.

Позиционное значение (place-value) – значимость цифры в ряде, в котором последовательные цифры – это множители последовательно возрастающих степеней основания.

Изобретение позиционной нумерации, основанной на значении цифр в зависимости от их положения в обозначении числа, как полагают, было сделано шумерами[1]; эта система нумерации была развита индусами, и она имеет большое значение для человечества. Очень полезное свойство

[1] Шумер - одна из древнейших цивилизаций Ближнего Востока, существовавшая в конце 4 - начале 2 тыс. до н. э. в Южном Двуречье, области нижнего течения Тигра и Евфрата, на юге современного Ирака.

позиционной системы состоит в том, что все числа, и малые и сравнительно большие, могут быть записаны с помощью небольшого числа различных цифр.

В вычислительных системах часто применяется двоичная (binary) система счета, в которой каждое число выражается при помощи двух цифр

0 и 1,

а основание этой системы равно 2.

С теоретической точки зрения система счисления, построенная по позиционному принципу с основанием 2, выделяется в том смысле, что это основание – наименьшее возможное.

Принцип двоичного счисления впервые был сформулирован в 17 веке Лейбницем[2]. Лейбниц высоко оценивал двоичную систему – в бинарной арифметике Лейбниц видел прообраз творения. Ему представлялось, что единица представляет божественное начало, а нуль – небытие, и что Высшее Существо создает все сущее из небытия точно таким же образом, как единица и нуль в его системе выражают все числа.

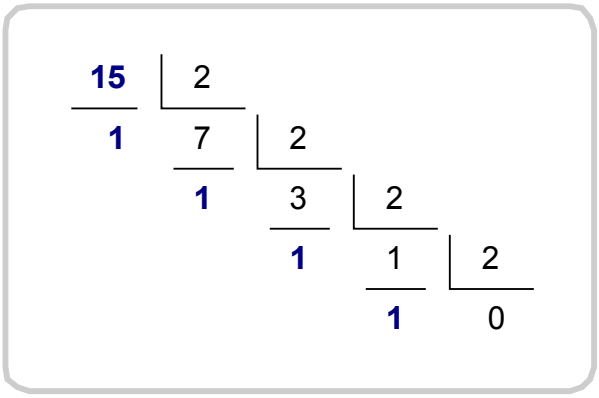
Эта система – основа современных цифровых вычислений. Для обозначения двоичных цифр применяется термин бит – сокращение bit английского словосочетания binary digit (двоичная цифра).

Применяя запись, аналогичную записи десятичных чисел, получим, например:

$$\begin{aligned} 1111 \text{ (двоичное число)} &= 1\times2^3 + 1\times2^2 + 1\times2^1 + 1\times2^0 \\ &= 1\times8 + 1\times4 + 1\times2 + 1\times1 \\ &= 15 \text{ (десятичное число).} \end{aligned}$$

Выполнив последовательно целочисленное деление десятичного числа 15 на степени основания двоичной системы, получим целые остатки от деления – двоичное представление числа 15:

[2] Лейбниц (Leibniz) Готфрид Вильгельм (1646-1716) – немецкий философ, математик. Развил учение о прирожденной способности ума к познанию высших категорий бытия, всеобщих и необходимых истин логики и математики. Один из создателей дифференциального и интегрального исчислений.



Общее правило для перехода от основания 10 к любому основанию b:

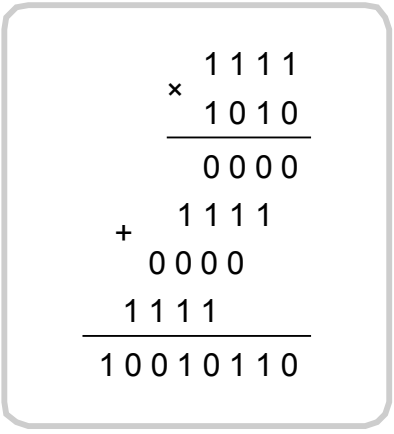
нужно выполнять последовательное деление на основание b, начиная с данного числа; целые остатки и будут "цифрами" для записи числа в системе с основанием b.

Подобно десятичной системе, в двоичной системе также можно выделить разряды: разряд единицы ($2^0 = 1$), двух (2^1), четырех (2^2), восьми (2^3), ...

Отличие этой системы в том, что в каждом разряде могут быть только два значения – 1 или 0, и соответственно, разряд либо вносит, либо не вносит свой вклад в суммарную величину.

Чтобы проиллюстрировать умножение в двоичной системе, умножим число 15 на 10, которые в двоичной записи имеют вид 1111 и 1010, соответственно.

Принимая во внимание, что в двоичной системе 1 + 1 = 10 (перенос единицы в следующий разряд), запишем:



и в итоге получается 150 (= 128+0+0+16+0+4+2+0).

Когда нужно указать, в какой системе записано число, основание записывают, например, нижним (подстрочным) индексом:

$1111_2 = 15_{10}$.

В математике и в вычислениях, шестнадцатеричная система (hexadecimal, hex, base-16) – система счета с основанием 16. В этой системе используются шестнадцать символов (цифр):

0, 1, 2, 3, 4 , 5 , 6 , 7 , 8 , 9

– значения от нуля до девяти и

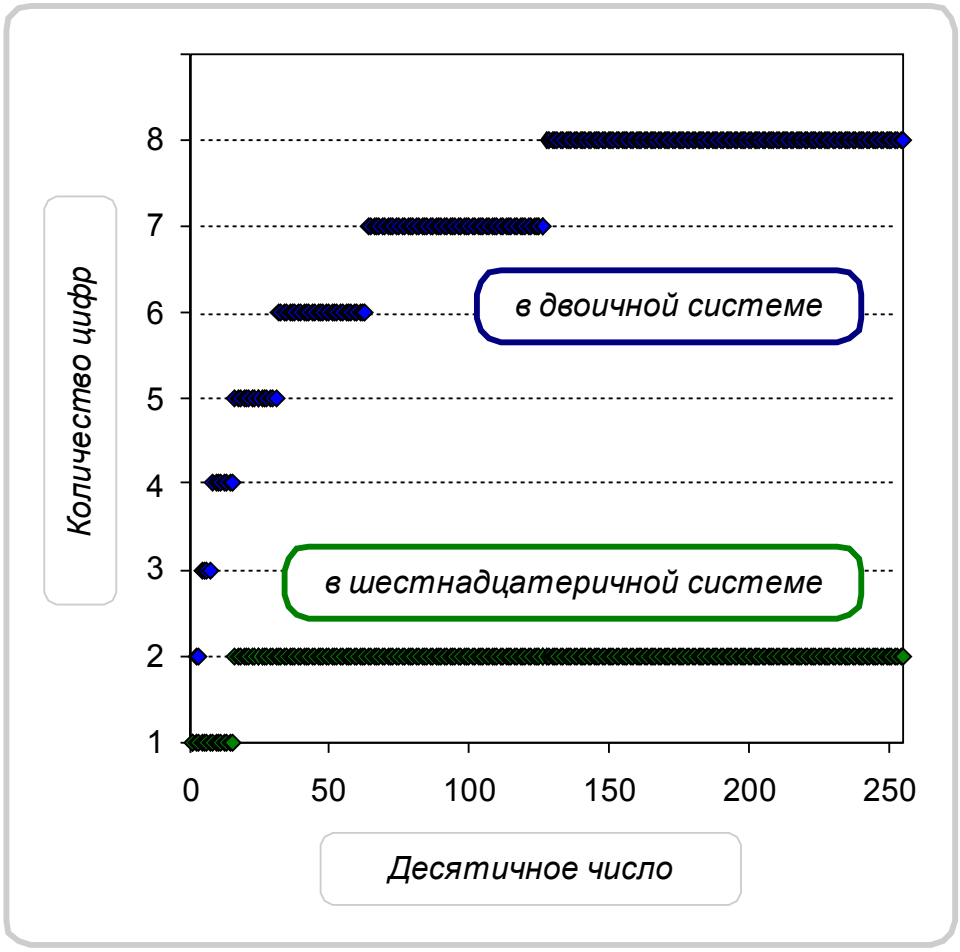
A, B, C, D, E, F (или a, b, c, d, e, f)

– соответствуют десятичным числам от 10 до 15.

Часто байт (8 бит) отображается двумя полубайтами. Полубайт (или нибл, от nibble) – группа из четырех битов, – может представить 16 возможных значений (2^4 – каждый бит может принимать два значения, битов четыре). Тогда полный 8-битовый байт представляется двумя шестнадцатеричными цифрами.

В языках программирования для обозначения системы счета вместе с числом пишут дополнительные знаки. В Turbo Pascal на шестнадцатеричное число указывает стоящий перед числом знак доллара ($), например, $9F.

На следующем рисунке сравнивается количеств цифр, необходимых для записи в двоичной и шестнадцатеричной системах десятичных чисел от 0 до 255.



Определенная непрактичность двоичной системы очевидна: для записи чисел нужны более длинные последовательности цифр.

Для перехода к десятичному представлению, число другой системы счета представляется полиномом

Вернемся к представлению десятичного числа в виде суммы произведений цифр на последовательные степени основания системы, 10. Запишем это представление в форме общей формулы.

Обозначим различные коэффициенты (цифры) одной и той же буквой, но с разными индексами (номерами)

$$a_0, a_1, a_2, \ \ldots\ , a_{n-2}, a_{n-1}, a_n\ ,$$

а наибольшую степень числа 10 обозначим как 10^n, понимая под n произвольное неотрицательное целое число. Тогда любое целое число в десятичной системе может быть представлено в виде суммы слагаемых

$$a_n \times 10^n + a_{n-1} \times 10^{n-1} + a_{n-2} \times 10^{n-2} + \ \ldots\ + a_1 \times 10 + a_0$$

и записано в символьной форме

$$a_n\, a_{n\text{-}1}\, a_{n\text{-}2}\ \ .\ .\ .\ \ a_2\, a_1\, a_0.$$

Как и в примере выше, коэффициенты (цифры) a_n, $a_{n\text{-}1}$, $a_{n\text{-}2}$, . . . , a_2, a_1, a_0 – это целые остатки, полученные при последовательном делении числа на основание системы, 10.

Аналогично рассматривая представление двоичного числа в виде суммы произведений цифр на последовательные степени основания двоичной системы, запишем это представление в виде общей формулы.

Обозначим различные коэффициенты одной и той же буквой, но с разными индексами (номерами)

$$a_0,\ a_1,\ a_2,\ \ .\ .\ .\ ,\ a_{n\text{-}2},\ a_{n\text{-}1},\ a_n,$$

а наибольшую степень числа 2 обозначим как 2^n, понимая под n произвольное неотрицательное целое число. Тогда любое целое число в двоичной системе может быть представлено в виде суммы

$$a_n \times 2^n + a_{n-1} \times 2^{n-1} + a_{n-2} \times 2^{n-2} + \ .\ .\ . + a_1 \times 2 + a_0$$

и записано в символьной форме

$$a_n\, a_{n\text{-}1}\, a_{n\text{-}2}\ \ .\ .\ .\ \ a_2\, a_1\, a_0.$$

Цифры a_n, $a_{n\text{-}1}$, $a_{n\text{-}2}$, . . . , a_2, a_1, a_0 – это также целые остатки, но полученные при последовательном делении числа на основание 2.

Формула для общего случая произвольного основания b:

$$a_n \times b^n + a_{n-1} \times b^{n-1} + a_{n-2} \times b^{n-2} + \ .\ .\ . + a_1 \times b + a_0$$

Полученные формулы представления десятичных и двоичных чисел в виде сумм – это полиномы (или функции) соответствующих оснований систем счета.

Полином (polynomial, от греч. polys – многочисленный, обширный и лат. nomen – имя) или **многочлен** – математическое выражение, состоящее из суммы конечного числа отдельных слагаемых (одночленов), каждое из которых является произведением константы (постоянной величины) и одной или более переменных величин, возведенных в неотрицательную целую степень.

Другими словами, полином в математике – выражение конечной длины, построенное из переменных величин (или неизвестных величин) и констант, используя для переменных операции сложения, вычитания и умножения.

Например, $2 \times x^2 + 7 \times x - 3$ – полином, а $2 \times x^{2/3} + 7 / x - 3$ – нет, потому что его первое слагаемое содержит нецелочисленный показатель степени, а во втором выполняется деление на x (отрицательный показатель степени).

Полином одного переменного числа x можно записать в виде суммы:

$$f(x) = a_0 + a_1 \times x + a_2 \times x^2 + \ \ldots \ + a_{n-2} \times x^{n-2} + a_{n-1} \times x^{n-1} + a_n \times x^n,$$

где

a_0, a_1, a_2, $\ \ldots \ $, a_{n-2}, a_{n-1}, a_n – постоянные коэффициенты многочлена, n – постоянное целое неотрицательное число.

В математике алгебраическая сумма слагаемых обозначается прописной (заглавной) греческой буквой Σ (сигма), и запись суммы имеет сокращенный вид:

$$\sum_{k=0}^{n} a_k x^k = a_0 + a_1 x + a_2 x^2 + \ldots + a_{n-2} x^{n-2} + a_{n-1} x^{n-1} + a_n x^n$$

Схема Горнера – алгоритм вычисления произвольного полинома

Схема Горнера или **алгоритм Горнера** (названы по имени Уильяма Горнера[3]) – это алгоритм для эффективного вычисления полиномов в форме отдельных алгебраических выражений, включающих переменное число.

Вычисление значения полинома с использованием схемы Горнера показано ниже для случая полинома третьей степени и обобщено для произвольной степени полинома в реализации на Pascal:

[3] Уильям Джордж Горнер (1786 – 1837) – британский математик.

Полином третьей степени:

$$f(x) = a_0 + a_1 \cdot x + a_2 \cdot x^2 + a_3 \cdot x^3$$

Вычисление:

схема Горнера	*псевдокод*
$f = a_3$	`f := a[3]`
$f = f \cdot x + a_2$	`f := f*x + a[2]`
$f = f \cdot x + a_1$	`f := f*x + a[1]`
$f = f \cdot x + a_0$	`f := f*x + a[0]`

реализация на Pascal

```
f := a[High(a)];
for i := High(a)-1 DownTo Low(a) do
  f := f * x + a[i];
```

Этот алгоритм имеет то преимущество, что при его применении число операций в вычислении сокращается, в сравнении с прямым способом вычисления.

Схема Горнера применяется, например, в процедурах перехода от одной системы счета к другой. Тогда, для полинома одной переменной величины, постоянные коэффициенты полинома – цифры определенной системы счета, а переменное число полинома – основание этой системы счета.

Записываем числа в разных системах счета

В последующем изложении будем применять некоторые определения.

Операция в вычислениях – выполнение вычислительной системой какого-либо действия над исходными величинами, включая их передачу, по одной из инструкций программы. Различают, например, операции математические, логические и др.

Оператор (operator) – математическое понятие, означающее соответствие между элементами двух множеств. Оператор каждому элементу a из множества A ставит в соответствие некоторый элемент b из множества B. Аналогичный смысл имеют термины "преобразование", "функция".

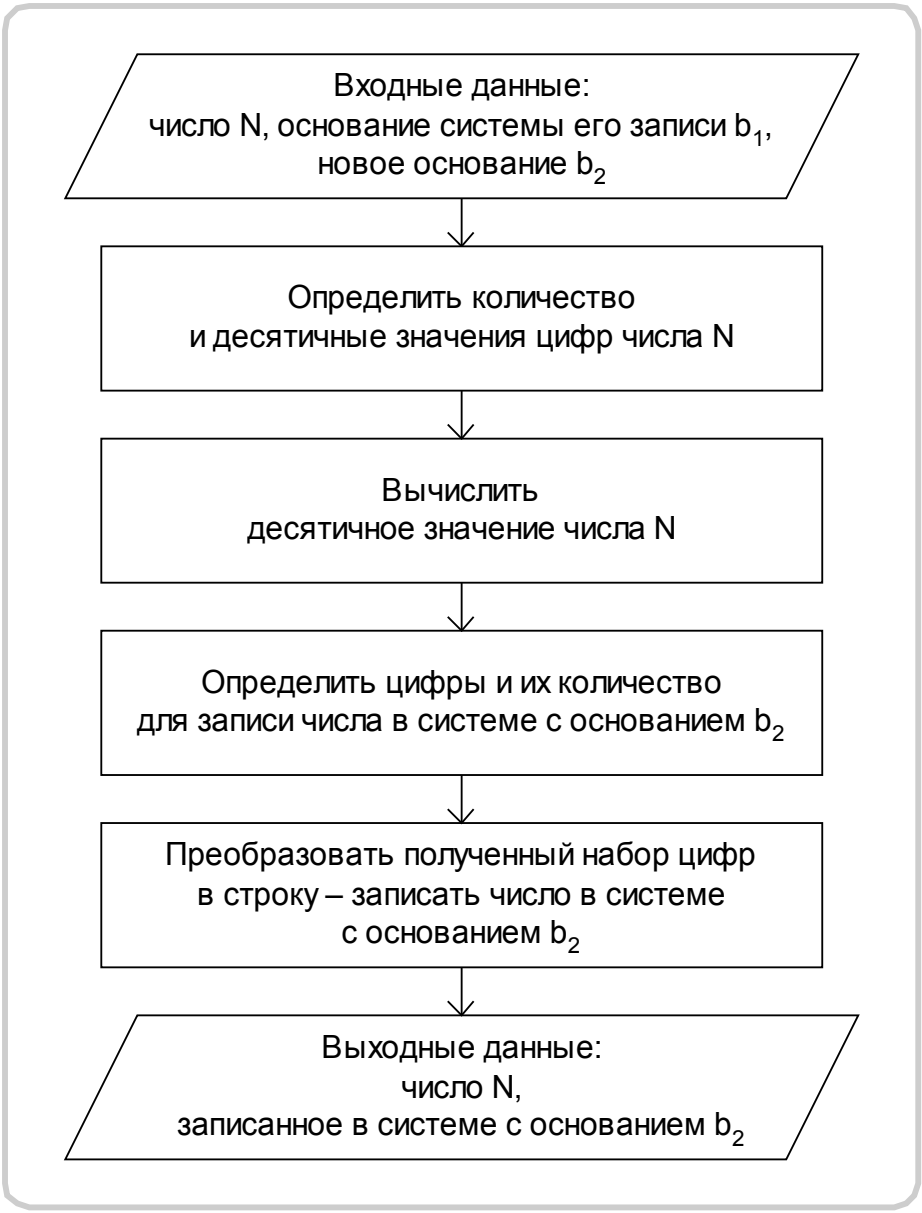
Составной оператор, составная инструкция (compound statement) в программировании – оператор, который может содержать в себе другие операторы. Такие операторы используются в ситуациях, когда язык программирования допускает в конструкции применение только одного оператора, например, в операторах `if`, `while` и т. п.

Операнд (operand) – объект действия, величина или функция, над которой выполняется математическая или логическая операция.

Для перевода записи какого-либо числа из одной системы счета в другую, напишем процедуру на Turbo Pascal.

Если до начала практического воплощения сформулировать основные части алгоритма или набросать небольшую схему, то последовательность действий будет яснее, а времени будет затрачено меньше.

В нашем случае такой общей схемой может быть, например, следующая.



В некоторых процедурах программы будем применять бинарные операторы `Div` и `Mod` языка программирования Pascal. **Бинарными** они называются потому, что соответствующие операторам действия выполняются с двумя исходными величинами.

Действие оператора `Div` – целочисленное деление.

Значение `m Div n` – величина `m/n`, округленная в направлении нуля до ближайшего целого числа.

Результат действия оператора `Mod` – целый остаток от деления.

Оператор `Mod` возвращает остаток от деления, полученный, делением его операндов.

Соотношение между результатами операций `Div` и `Mod` следующее:

```
m mod n = m - (m div n) * n.
```

В следующей программе десятичные коды некоторых букв русского алфавита записываются в двоичной системе. Результаты сохраняются в файл "RuChars.txt ".

```
program RuChars;

uses
  Mix;

var
  i: Integer;
  InputNumber, OutputNumber: String;
  OutputF: Text;
begin
    Assign(OutputF, 'RuChars.txt');
    Rewrite(OutputF);
    for i := 192 to 196 do
    begin
      Str(i, InputNumber);
      ChangeNumberBase(InputNumber, 10, 2, OutputNumber);
      Writeln(OutputF, '  The decimal code ', i:3,
        ' for "', Chr(i), '"', ' in binary system is ',
        OutputNumber);
    end;
    Close(OutputF);
end.
```

```
  The decimal code 192 for "А" in binary system is 11000000
  The decimal code 193 for "Б" in binary system is 11000001
  The decimal code 194 for "В" in binary system is 11000010
  The decimal code 195 for "Г" in binary system is 11000011
  The decimal code 196 for "Д" in binary system is 11000100
```

Программа "B10_2_16.PAS", приведенная ниже, выполняет запись десятичного числа 2147483647 (=2^31-1) в двоичной и шестнадцатеричной системах счета. Результаты сохраняются в файле "B10_2_16.TXT" и приведены ниже.

```
program B10_2_16;

uses
  Mix;

var
  NBase10, NBase2, NBase16: String;
  OutputF: Text;

begin
  Assign(OutputF, 'B10_2_16.txt');
  Rewrite(OutputF);

  Str(High(ULongInt), NBase10);

  ChangeNumberBase(NBase10, 10,  2, NBase2);
  ChangeNumberBase(NBase10, 10,  16, NBase16);

  Writeln(OutputF, NBase2,  '[2] =');
  Writeln(OutputF, NBase10, '[10] =');
  Writeln(OutputF, NBase16, '[16]');

  Close(OutputF);
end.
```

```
            11111111111111111111111111111111[2] =
                      2147483647[10] =
                        7FFFFFFF[16]
```

В следующей программе выполняется запись неотрицательных чисел до 31 в системах счета с основаниями 2, 3, 5, 7, 8, 16. Результаты сохраняются в файле "NUMBERS.TXT" и приведены ниже в форме таблицы.

```
program NumSys;

uses
  Mix;

var
  i: Byte;
  NBase10,
  NBase2, NBase3, NBase5,
  NBase7, NBase8, NBase16: String;
  OutputF: Text;
```

```
begin
  Assign(OutputF, 'Numbers.txt');
  Rewrite(OutputF);
  Writeln(OutputF, 'Base=10 Base=2 Base=3 Base=5 ',
    'Base=7 Base=8 Base=16');
  for i := 0 to 31 do
  begin
    Str(i, NBase10);
    ChangeNumberBase(NBase10, 10,  2, NBase2);
    ChangeNumberBase(NBase10, 10,  3, NBase3);
    ChangeNumberBase(NBase10, 10,  5, NBase5);
    ChangeNumberBase(NBase10, 10,  7, NBase7);
    ChangeNumberBase(NBase10, 10,  8, NBase8);
    ChangeNumberBase(NBase10, 10, 16, NBase16);
    Writeln(OutputF, NBase10, ' ', NBase2, ' ', NBase3, ' ',
      NBase5, ' ', NBase7, ' ', NBase8, ' ', NBase16);
  end;
  Close(OutputF);
end.
```

Base=10	Base=2	Base=3	Base=5	Base=7	Base=8	Base=16
0	0	0	0	0	0	0
1	1	1	1	1	1	1
2	10	2	2	2	2	2
3	11	10	3	3	3	3
4	100	11	4	4	4	4
5	101	12	10	5	5	5
6	110	20	11	6	6	6
7	111	21	12	10	7	7
8	1000	22	13	11	10	8
9	1001	100	14	12	11	9
10	1010	101	20	13	12	A
11	1011	102	21	14	13	B
12	1100	110	22	15	14	C
13	1101	111	23	16	15	D
14	1110	112	24	20	16	E
15	1111	120	30	21	17	F
16	10000	121	31	22	20	10
17	10001	122	32	23	21	11
18	10010	200	33	24	22	12

19	10011	201	34	25	23	13
20	10100	202	40	26	24	14
21	10101	210	41	30	25	15
22	10110	211	42	31	26	16
23	10111	212	43	32	27	17
24	11000	220	44	33	30	18
25	11001	221	100	34	31	19
26	11010	222	101	35	32	1A
27	11011	1000	102	36	33	1B
28	11100	1001	103	40	34	1C
29	11101	1002	104	41	35	1D
30	11110	1010	110	42	36	1E
31	11111	1011	111	43	37	1F

Двоичные числа хранятся, обрабатываются и передаются битами, байтам, машинными словами

Бит (bit, сокращение binary digit) – базисная единица элемента данных в вычислениях, в коммуникационных сетях дистанционной передачи данных.

Это количество данных, которое может быть обозначено или сохранено физически устройством, материальной системой, в которых выделены только два различных и измеримых устойчивых состояния.

Например, два положения электрического выключателя "включено" и "выключено" могут соответствовать двум устойчивым состояниям для обозначения единицы информации.

Материальные системы, описываемые другими закономерностями, могут иметь иные стабильные состояния для обозначения единиц информации. Биту может быть поставлен в соответствие любой признак системы, который может принимать два значения.

В некоторых типах постоянных запоминающих устройств бит может быть представлен наличием или отсутствием токопроводящей дорожки в некоторой точке микросхемы.

В полупроводниковом запоминающем устройстве, как, например, в устройстве с произвольной выборкой данных (в динамической оперативной памяти) или в электрически перепрограммируемом постоянном запоминающем устройстве (флэш-памяти), два значения бита могут быть

представлены двумя уровнями электрического заряда, накопленного в конденсаторе.

В оптических дисках, бит закодирован как наличие или отсутствие темного, не отражающего микроскопического углубления на отражающей поверхности диска.

В микросхемах с двумя устойчивыми состояниями значениями бита могут быть два разных уровня напряжения или электрического тока, допускаемых микросхемой. В вычислительных системах бит обычно представляется электрическим напряжением, импульсом тока или состоянием триггерной схемы[4].

Соответствие между значениями бита и состояниями лежащего в основе устройства – вопрос соглашения, и для какой-либо величины может быть использовано разное присваивание значения, даже в пределах одного устройства или одной программы.

В вычислениях, бит (двоичная цифра), определяется как переменная или вычисляемая величина, которая может принимать только два возможных значения. Эти два значения интерпретируются как двоичные цифры и обычно обозначаются арабскими цифрами `0` и `1`. Эти два значения могут также интерпретироваться как логические значения `true` или `false` (истина или ложь), алгебраические знаки + или −, состояния системы "включено" или "выключено". Во многих распространенных языках программирования число `0` преобразуемо в логическое значение `false`, а `1` – в `true`.

В вычислениях имеются несколько единиц измерения данных, которые определяются как кратные числа битов.

Кратные числа битов:

Обозначение	Количество битов в десятичной системе	Количество битов в двоичной системе
kilobit, килобит, (kbit, Кбит)	10^3	2^{10}
megabit, мегабит (Mbit, Мбит)	10^6	2^{20}
gigabit, гигабит (Gbit, Гбит)	10^9	2^{30}
terabit, терабит (Tbit, Тбит)	10^{12}	2^{40}

[4] Триггер (trigger) – электронное устройство или схема, которая может принимать любое из двух устойчивых состояний равновесия приложением соответствующего внешнего импульса.

petabit, петабит (Pbit, Пбит)	10^{15}	2^{50}

Байт (byte) – единица измерения количества данных, содержащая несколько битов. Для обозначения байта применяются буквы английского (и русского) алфавитов "B" (и "Б"). Наиболее часто байт считается равным 8 битам и в этом случае он может принимать 256 (2^8) различных значений.

Если, например, байт применяется для представления целых чисел без знака, то соответствующие числовые значения расположены в интервале [0...255].

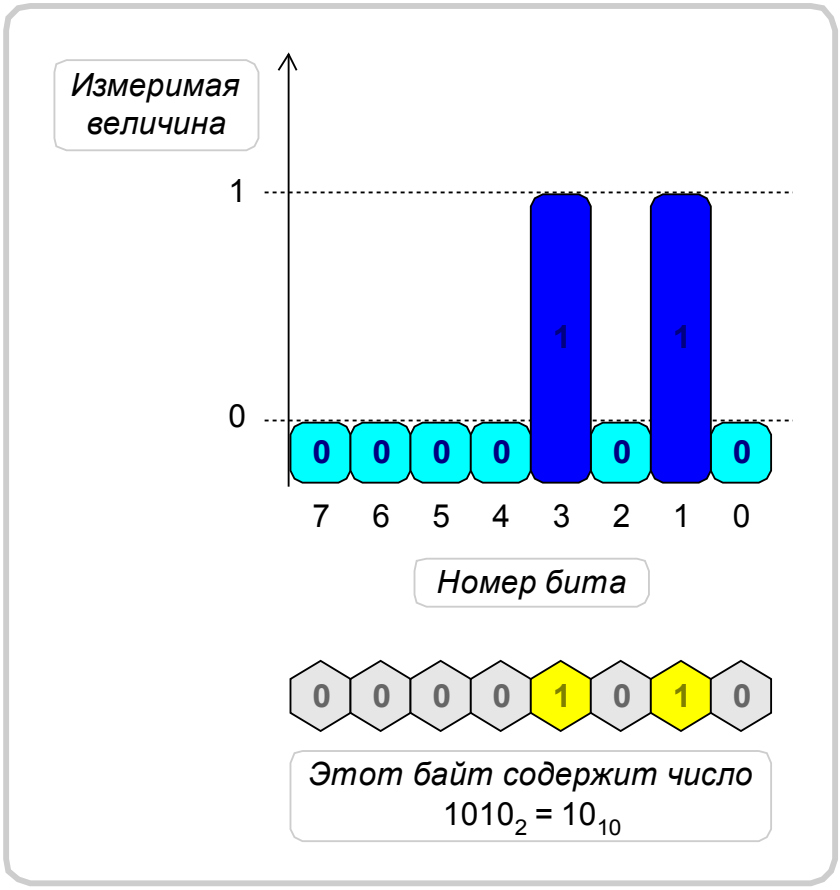
В вычислениях, **слово** (word) – строка битов, рассматриваемая как целостный элемент. Размер или длина слова – это число битов в слове.

Практически используемый в вычислениях размер слова может зависеть, например, от модели компьютера, операционной системы, используемого языка программирования и соответствующего компилятора[5].

В Turbo Pascal тип Word имеет размер 16 бит и описывает тип целых чисел без знака в интервале [0…65535].

Когда в вычислениях, или в языке программирования нужно указать определенный бит, расположенный в байте или в слове, то его положение определяется его номером от 0 до соответствующей позиции в пределах данной группы битов. Наиболее часто нумерация начинается с бита, представляющего младший разряд, как показано на следующем рисунке.

[5] Компилятор (compiler) – компьютерная программа (или набор программ), которая преобразует исходный код, написанный на языке программирования (source language) в другой целевой язык (target language), часто имеющий двоичную форму, известную как объектный код (object code). Преобразование выполняется с целью создать программу, которая может быть выполнена процессором.



Поразрядные операции – действия с битами байтов, двоичными числами

Отличие поразрядных операторов от логических операторов

Логическая операция (logic operation) – операция, аргументы которой принимают два логических значения: true (истина) или false (ложь).

В Pascal аргументы логических операций имеют тип Boolean. В программах такие аргументы (исходные величины) обрабатываются как целостные, отдельные величины, имеющие значения true или false, или изменяющие значения true на false и наоборот.

Поразрядные операции, в общем случае, производятся индивидуально над каждым битом какой-либо целой величины. По этой причине поразрядные операции обычно не применяются для записи условий; для этого используются логические операторы и операторы сравнения (отношения).

Поразрядные операторы AND, OR, XOR, NOT

С расположением бита в байте или в (машинном) слове связано его позиционное значение, или, другими словами, положение бита в байте соответствует определенному разряду байта. Программирование предоставляет возможность производить операции с отдельными битами, или разрядами исходной величины, значение которой хранится в байте или в слове.

Операторы, действие которых определено для индивидуальных битов байта, или битов находящихся в соответствующих разрядах байтов, называют **поразрядными операторами** (bitwise operators).

К поразрядным операциям относятся:

– тестирование (проверка значений битов),

– установка определенных значений битов,

– или смещение всех битов влево или вправо

в байте или в слове исходной величины.

Для того чтобы такие операции были возможны в вычислительной системе, данная исходная величина должна относиться к одному из целочисленных типов данных.

Целочисленные типы в Turbo Pascal – Shortint, Integer, Longint, Byte, Word. Поразрядные операции не могут быть применены к данным, имеющим тип Single, Real; Double, Extended или к другим, более сложным типам данных.

Будем рассматривать область *неотрицательных чисел*, ограниченную одним байтом – тип данных Byte.

Для описания действия поразрядных операторов возьмем два числа, например,

$90_{10} = 01011010_2$

и

$122_{10} = 01111010_2$.

Оператор NOT

Оператор отрицания NOT (НЕ) имеет один аргумент, это **унарный оператор** (unary). Он изменяет состояния всех битов байта-операнда на противоположные значения.

Например,

NOT 90	0	1	0	1	1	0	1	0
165	1	0	1	0	0	1	0	1

Оператор AND

Поразрядный оператор AND (И) возвращает значение 1 только в том случае, когда все соответствующие биты операндов равны 1. Например,

90 AND 122	0	1	0	1	1	0	1	0
	0	1	1	1	1	0	1	0
90	0	1	0	1	1	0	1	0

Действие оператора AND – умножение битов. То есть, если бит в любом из операндов равен 0, соответствующий бит результата получит значение 0.

Такое действие оператора AND – способ "выключения" битов или, установки нулевых значений в определенных битах результата. Эта операция также может быть применена, чтобы отделить часть строки битов.

В программировании оператор AND используется, например, для проверки значений определенных разрядов (является ли определенный бит 1 или 0).

Поразрядный оператор AND может быть объединен с оператором NOT, чтобы очистить определенные биты.

Оператор OR

Оператор OR (ИЛИ) устанавливает значение бита результата равным 1, когда равен 1 хотя бы один соответствующий бит любого из операндов. Например,

90 OR 122	0	1	0	1	1	0	1	0
	0	1	1	1	1	0	1	0
122	0	1	1	1	1	0	1	0

Оператор OR может быть использован для "включения" битов, или установки единиц в определенных битах результата.

Оператор XOR

Оператор XOR (exclusive OR) – исключающее ИЛИ, оператор неэквивалентности. Значение бита результата, получаемое в результате операции XOR равно 1 только тогда, когда значения соответствующих битов операндов не равны.

Например,

90 XOR 122	0	1	0	1	1	0	1	0
	0	1	1	1	1	0	1	0
32	0	0	1	0	0	0	0	0

Другими словами, бит результата равен 1, если только один из соответствующих битов аргументов равен 1; в других случаях бит результата равен 0.

Выполнение XOR для какого-либо значения с таким же значением всегда в результате дает нуль:

90 XOR 90	0	1	0	1	1	0	1	0
	0	1	0	1	1	0	1	0
0	0	0	0	0	0	0	0	0

Поразрядный оператор XOR может быть использован для переключения флагов[6] (признаков) в совокупности битов.

Этот способ может быть применен, чтобы управлять определенной комбинацией битов, которая представляет набор переменных, имеющих тип Boolean.

Для получения числовых значений примеров поразрядных операций была использована приведенная ниже программа. Получаемые результаты выводятся в текстовый файл "BITWISE.TXT".

```
program BitWise;

uses
  Mix;

var
  OutputF: Text;
  FileName: String;

  Ch1, Ch2: Char;
```

[6] Флаг в вычислительных системах – признак, который может быть установлен в определенное состояние или возвращен в исходное состояние, и применяемый, чтобы обозначить определенное условие или состояние, или служит как указание начала определенного ответного действия в выполнении программы вычислений.

```
  Number1, Number2, Number3: Byte;
  NumberBase10, Number1Base2,
  Number2Base2, Number3Base2: String;

  procedure ResultsOutput(OP: String; FileName: String);
  var
    OutF: Text;
  begin
    Str(Number3, NumberBase10);
    ChangeNumberBase(NumberBase10, 10,  2, Number3Base2);

    Assign(OutF, FileName);
    Append(OutF);

    Writeln(OutF, Number1, ' ', OP, ' ', Number2,
      ' = ', Number3);

    Writeln(OutF);
    Writeln(OutF, #9, Number1Base2);
    Writeln(OutF, OP, #9, Number2Base2);
    Writeln(OutF, #9, Number3Base2);
    Writeln(OutF);

    Close(OutF);
  end;

begin
  Ch1 := 'Z';
  Ch2 := 'z';
  Number1 := Ord(Ch1);
  Number2 := Ord(Ch2);

  Str(Number1, NumberBase10);
  ChangeNumberBase(NumberBase10, 10,  2, Number1Base2);
  Str(Number2, NumberBase10);
  ChangeNumberBase(NumberBase10, 10,  2, Number2Base2);

  FileName := 'BitWise.txt';

  Assign(OutputF, FileName);
  Rewrite(OutputF);

  Writeln(OutputF, 'Number1:  Ord(', Ch1,') = ', Number1:3,
    ' (decimal),  ', Number1Base2, ' (binary)');
  Writeln(OutputF, 'Number2:  Ord(', Ch2,') = ', Number2:3,
    ' (decimal),  ', Number2Base2, ' (binary)');
  Writeln(OutputF);

  Number3 := NOT Number1;
  Str(Number3, NumberBase10);
  ChangeNumberBase(NumberBase10, 10,  2, Number3Base2);
```

```
  Writeln(OutputF, 'NOT ', Number1, ' = ', Number3);
  Writeln(OutputF);
  Writeln(OutputF, 'NOT',#9, Number1Base2);
  Writeln(OutputF, #9, Number3Base2);

  Writeln(OutputF);
  Close(OutputF);

  Number3 := Number1 AND Number2;
  ResultsOutput('AND', FileName);

  Number3 := Number1 OR Number2;
  ResultsOutput('OR', FileName);

  Number3 := Number1 XOR Number2;
  ResultsOutput('XOR', FileName);
end.
```

```
Number1:  Ord(Z) =  90 (decimal),  01011010 (binary)
Number2:  Ord(z) = 122 (decimal),  01111010 (binary)

NOT 90 = 165

NOT  01011010
     10100101

90 AND 122 = 90

     01011010
AND  01111010
     01011010

90 OR 122 = 122

     01011010
OR   01111010
     01111010

90 XOR 122 = 32

     01011010
XOR  01111010
     00100000
```

Таблицы истинности поразрядных операций

В вычислениях, **значение истинности** (truth-value) – одно из значений, true (истина) или false (ложь), которое может принимать инструкция-утверждение.

В случае поразрядных операций значение истинности – это возможная величина b бита: b = 0 или b = 1.

Таблица истинности (truth table) – таблица, используемая в логике, и указывающая значение истинности составной инструкции-утверждения для каждого значения истинности ее составных частей-утверждений.

Другими словами, это таблица, полностью описывающая логическую функцию перечислением всех возможных комбинаций входных значений (сигналов) и соответствующие каждой комбинации значения (сигналы) на выходе.

Подобные таблицы используются в технологиях полупроводниковых устройств, чтобы указать значения выходных сигналов логической схемы для каждой комбинации значений входных сигналов.

Таблицы истинности операций NOT, AND OR XOR

NOT

	NOT a
a = 0	1
a = 1	0

AND

a AND b	b = 0	b = 1
a = 0	0	0
a = 1	0	1

OR

a OR b	b = 0	b = 1

a = 0	0	1
a = 1	1	1

XOR

a XOR b	b = 0	b = 1
a = 0	0	1
a = 1	1	0

Тип результата

В Pascal, следующие правила применяются к поразрядным операциям:

– тип результата операции NOT такой же, как тип операнда;

– тип результата операций AND, OR, XOR – целочисленный тип данных с областью значений, которая включает все возможные значения обоих типов операндов (но не более).

Сдвиг битов

В вычислениях, **сдвиг** (shift) – перемещение группы битов, содержащихся в регистре[7], на заданное число позиций влево или вправо.

Регистры в процессоре имеют определенное число битов, доступных для хранения последовательности цифр числа. Поэтому после сдвига, некоторые разряды хранившегося числа окажутся за пределами регистра, а с другой стороны регистра столько же битов добавятся.

Операции сдвига различаются, в зависимости от того, как вычисляются значения добавляемых битов.

[7] В вычислениях, регистры – устройства быстродействующей памяти в центральном процессоре, служащие для временного хранения команд, операндов и/или результатов выполняемых операций.
Центральный процессор исполняет логические и арифметические операции над исходными элементами данных, как определено командами.

Сдвиг битов в Turbo Pascal

Ограничимся операциями сдвига целых чисел без знака. Такие числа в Pascal соответствуют типам Byte, Word.

Для выполнения сдвига битов применяются операторы: `ShL` – сдвиг битов влево и `ShR` – сдвиг битов вправо.

Операции

```
m ShL n
```

и

```
m ShR n
```

сдвигают значение `m` влево или вправо на `n` бит.

Если `m` – целое число без знака, операция сдвига битов влево на `n` эквивалентна умножению `m` на 2^n, при условии, что не происходит переполнение[8] результата.

Операция сдвига `m` вправо на `n` эквивалентна делению m на 2^n.

Тип результата операции сдвига числа `m` на количество позиций `n` такой же, какой тип у числа `m`.

Например, если `m` содержит значение 01011010 (десятичное 90), а `n` равно 1, операция

90 ShL 1	0	1	0	1	1	0	1	0
180	1	0	1	1	0	1	0	0

возвращает 10110100, десятичное 180 (= $2^7 + 0 + 2^5 + 2^4 + 0 + 2^2 + 0 + 0$).

Следующая программа иллюстрирует действие операторов сдвига. Когда возвращаемый результат располагается за диапазоном определенного типа данных, выводится сообщение о переполнении – "RE".

[8] В вычислениях, переполнение (overflow, arithmetic overflow) – состояние, которое происходит, когда числовые операции производят результаты слишком большие, чтобы сохранить их в области памяти, выделенной для этого.

```
program IntShift;

uses
  Mix;

var
  b1, bs: Byte;
  w1: Word;
  i: 0..31;
  l1, tmp: ULongInt;
  RangeError: Boolean;

begin
  { The operations (m SHL n) and (m SHR n)
    shift the value of m to the left or right by n bits.
    ( n = n mod SizeOf(type of m) )

    If m is an unsigned integer:
       (m SHL n) = m * 2^n;
       (m SHR n) = m / 2^n.
  }

  b1 := Low(Byte) + 1;
  w1 := Low(Word) + 1;
  l1 := 1;

  bs := 1;

  Writeln('Shift values to the left: ',
    'Byte, Word, ULongInt.');
  Writeln('Initial values:');

  Writeln('Byte = ', b1, ',  Word = ', w1,
    ',  ULongInt = ', l1, '.');
  Writeln;

  Writeln('ShL by', #9#9, 'Byte', #9, 'Word', #9,
    'ULongInt');
  for i := 0 to 15 do
  begin
    Write(bs:2, #9);

    SHLWithRangeCheck(b1, Low(Byte), High(Byte), bs,
      tmp, RangeError);
    if RangeError then
      Write(#9, 'RE')
    else
      Write(#9, tmp);
```

```
    SHLWithRangeCheck(w1, Low(Word), High(Word), bs,
      tmp, RangeError);
    if RangeError then
      Write(#9, 'RE')
    else
      Write(#9, tmp);

    SHLWithRangeCheck(l1, Low(ULongInt), High(ULongInt),
      bs, tmp, RangeError);
    if RangeError then
      Writeln(#9, 'RE')
    else
      Writeln(#9, tmp);

    bs  := bs + 1;
  end;

  Readln;
  Exit;
end.
```

```
Shift values to the left: Byte, Word, ULongInt.
Initial values:
Byte = 1,  Word = 1,  ULongInt = 1.

ShL by          Byte    Word    ULongInt
 1              2       2       2
 2              4       4       4
 3              8       8       8
 4              16      16      16
 5              32      32      32
 6              64      64      64
 7              128     128     128
 8              RE      256     256
 9              RE      512     512
10              RE      1024    1024
11              RE      2048    2048
12              RE      4096    4096
13              RE      8192    8192
14              RE      16384   16384
15              RE      32768   32768
16              RE      RE      65536
```

В следующей программе с помощью операций сдвига и получения целочисленных остатков, буквам английского алфавита ставятся в соответствие другие отображаемые символы кодовой страницы.

```
program ChrShift;

var
  i, bs: Byte;
  b1, b2, b3: Byte;
  Ch1, Ch2, Ch3: Char;
  OutF: Text;

begin
  Ch1 := 'A';
  Ch2 := 'Z';
  b1 := Ord(Ch1);
  b2 := Ord(Ch2);

  bs := 3;

  Assign(OutF, 'ChrShift.txt');
  Rewrite(OutF);
  Writeln(OutF, 'Operators ShL and Mod;',
    ' return Byte and Char values:');
  Writeln(OutF);
  for i := b1 to b2 do
  begin
    b3 := ((i SHL bs) Mod 211) + 32;

    Ch1 := Chr(i);
    Ch3 := Chr(b3);

    Writeln(OutF, 'i=', i:3, #9, 'Chr(', i:3, ')=', Ch1:2,
     #9, '((', i:3, ' SHL ', bs, ') Mod 211) + 32 = ', b3:3,
     #9, 'Chr(', b3:3, ')=', Ch3:2);
  end;
  Close(OutF);
end.
```

Operators ShL and Mod; return Byte and Char values:

i= 65	Chr(65)= A	((65 SHL 3) Mod 211) + 32 = 130	Chr(130)= ,
i= 66	Chr(66)= B	((66 SHL 3) Mod 211) + 32 = 138	Chr(138)= Љ
i= 67	Chr(67)= C	((67 SHL 3) Mod 211) + 32 = 146	Chr(146)= ’
i= 68	Chr(68)= D	((68 SHL 3) Mod 211) + 32 = 154	Chr(154)= љ
i= 69	Chr(69)= E	((69 SHL 3) Mod 211) + 32 = 162	Chr(162)= ў
i= 70	Chr(70)= F	((70 SHL 3) Mod 211) + 32 = 170	Chr(170)= Є

i= 71	Chr(71)= G	((71 SHL 3) Mod 211) + 32 = 178	Chr(178)= I
i= 72	Chr(72)= H	((72 SHL 3) Mod 211) + 32 = 186	Chr(186)= є
i= 73	Chr(73)= I	((73 SHL 3) Mod 211) + 32 = 194	Chr(194)= В
i= 74	Chr(74)= J	((74 SHL 3) Mod 211) + 32 = 202	Chr(202)= К
i= 75	Chr(75)= K	((75 SHL 3) Mod 211) + 32 = 210	Chr(210)= Т
i= 76	Chr(76)= L	((76 SHL 3) Mod 211) + 32 = 218	Chr(218)= Ъ
i= 77	Chr(77)= M	((77 SHL 3) Mod 211) + 32 = 226	Chr(226)= в
i= 78	Chr(78)= N	((78 SHL 3) Mod 211) + 32 = 234	Chr(234)= к
i= 79	Chr(79)= O	((79 SHL 3) Mod 211) + 32 = 242	Chr(242)= т
i= 80	Chr(80)= P	((80 SHL 3) Mod 211) + 32 = 39	Chr(39)= '
i= 81	Chr(81)= Q	((81 SHL 3) Mod 211) + 32 = 47	Chr(47)= /
i= 82	Chr(82)= R	((82 SHL 3) Mod 211) + 32 = 55	Chr(55)= 7
i= 83	Chr(83)= S	((83 SHL 3) Mod 211) + 32 = 63	Chr(63)= ?
i= 84	Chr(84)= T	((84 SHL 3) Mod 211) + 32 = 71	Chr(71)= G
i= 85	Chr(85)= U	((85 SHL 3) Mod 211) + 32 = 79	Chr(79)= O
i= 86	Chr(86)= V	((86 SHL 3) Mod 211) + 32 = 87	Chr(87)= W
i= 87	Chr(87)= W	((87 SHL 3) Mod 211) + 32 = 95	Chr(95)= _
i= 88	Chr(88)= X	((88 SHL 3) Mod 211) + 32 = 103	Chr(103)= g
i= 89	Chr(89)= Y	((89 SHL 3) Mod 211) + 32 = 111	Chr(111)= o
i= 90	Chr(90)= Z	((90 SHL 3) Mod 211) + 32 = 119	Chr(119)= w

Простое поразрядное шифрование

Оператор неэквивалентности XOR обладает свойством, которое позволяет простым способом преобразовать сообщение из формы читаемого текста в код, то есть выполнить кодирование сообщения.

Двукратное применение поразрядной операции XOR к некоторой величине с одним и тем же вторым операндом дает исходную величину.

То есть,

$$c = a \text{ XOR } b,$$
$$a = c \text{ XOR } b$$

или

```
i[1] = (i[1] XOR i[2]) XOR i[2]
```

Таблица истинности этих операций:

(a XOR b) XOR b	b = 0	b = 1
a = 0	0	0
a = 1	1	1

Таким образом, результат двух последовательных поразрядных операций XOR с применением второго операнда одной величины – первоначальная величина.

Этот принцип можно использовать в небольшой программе для кодирования – преобразования сообщения, текста из читаемого текста в код, и декодирования – предобразования сообщения, текста из кода в читаемый текст, на обычный язык.

Для кодирования текста, в качестве ключа выбирается некоторое целое число, и выполняются операции XOR с символами текстового сообщения и выбранным числом-ключом; на выходе – зашифрованный текст.

Для декодирования зашифрованного текста повторно выполняются операции XOR с символами зашифрованного сообщения (с тем же самым ключом); на выходе – читаемый текст. Результат двух операций XOR использующих тот же самый ключ – исходное текстовое сообщение.

Возвращаясь к тексту, что далее

Возвращаясь к тексту сообщения, кратко обобщим алгоритм кодирования сообщения с использованием операции XOR на примере.

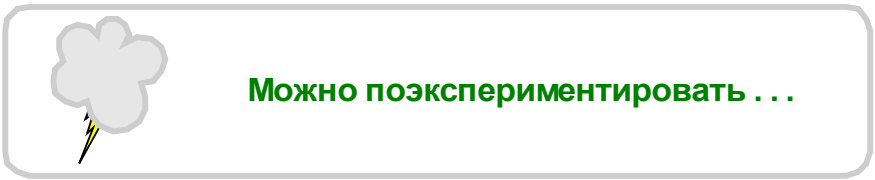
Для примера возьмем строку букв "Слово"; в качестве ключа воспользуемся буквой "А" (в кодовой странице CP1251 десятичный код равен 65, а в двоичной системе число записывается как 01000001).

Последовательно выполняемые операции с каждой буквой строки и выбранным ключом сведены с следующую таблицу.

С	209	11010001 XOR 01000001	10010000	144	ђ
л	235	11101011 XOR 01000001	10101010	170	Є
о	238	11101110 XOR 01000001	10101111	175	Ї
в	226	11100010 XOR 01000001	10100011	163	Ј
о	238	11101110 XOR 01000001	10101111	175	Ї

Реализация этой процедуры приведена в модуле XORCrypt.pas.

Пора поставить точку. Но на самом деле . . . точку здесь ставить рано.



Например, сохраняя текст в числовом формате (как бы данные), или используя оператор ShL, как в одной из программ выше.

Или, будут ли аналогично работать поразрядные операторы, приведенные в следующей таблице истинности (?):

a OR (a AND b)	b = 0	b = 1

a = 0	0	0
a = 1	1	1

Или другая группа операторов?

Те, кому это интересно, смогут придумать алгоритмы лучше и получить результат более надежный – зашифрованный текст, который трудно, если вообще возможно, расшифровать.

Удачи начинающим!

Используемый модуль XORCrypt.pas

```
unit XORCrypt;

interface
uses
{ List of unit dependencies goes here... }
  Mix;

var
    Ch: Char;
    ParStr: String;
    CryptoKey: Integer;
    InputFileName, OutputFileName: String;
    Msg: String;

  procedure XORTextCrypto(InputFileName,
    OutputFileName: String);
  {  XORTextCrypto (xor text cryptography) functions:
       1. Encode - to convert (message, text) from plain text
                   into code.
       2. Decode - to convert from code
                   into ordinary language (message, text).
  }

implementation
{ uses
  List of unit dependencies goes here... }

  procedure XORTextCrypto(InputFileName,
    OutputFileName: String);
  {  XORTextCrypto (xor text cryptography) functions:
       1. Encode - to convert (message, text) from plain text
                   into code.
       2. Decode - to convert from code
                   into ordinary language (message, text).
  }
  var
    InputF, OutputF: Text;
    InputString, OutputString: String;
    CharCode: Integer;
    i, len: Integer;
    IOCode: Integer;
  begin
    Assign(InputF, InputFileName);
```

```
    {$I-} { Disables code that checks the result
            of an I/O procedure }
    Reset(InputF);
    {$I+} { Enables code that checks the result
            of an I/O procedure }
    IOCode := IOResult;
    if IOCode = 0 then
    begin
      Assign(OutputF, OutputFileName);
      Rewrite(OutputF);
      while not(Eof(InputF)) do
      begin
        Readln(InputF, InputString);
        len := Length(InputString);
        OutputString := '';
        for i := 1 to len do
        begin
          CharCode := Ord(InputString[i]);
          OutputString := OutputString
            + Chr(CharCode XOR CryptoKey);
        end;
        Writeln(OutputF, OutputString);
      end;
      Close(InputF);
      Close(OutputF);
    end
    else
    begin
      Str(IOCode, Msg);
      Msg := InputFileName + ' : I/O error, code ' +
        Msg + '.';
      WaitForExit(Msg);
    end;
  end;

end.
```

Используемый модуль Mix.pas

```
{*********************************************************}
{ All numbers are unsigned integers in the unit 'Mix.pas' }
{*********************************************************}
unit Mix;

interface
{ uses
  List of unit dependencies goes here... }

  const
    AllDigits: array[0..15] of Char
      = ('0','1','2','3','4','5','6','7','8','9','A','B','C',
         'D','E','F');

  type
    { Unsigned: - not having a plus or minus sign,
                - not having a bit representing
                  a plus or minus sign }

    { Base: the number of distinct single-digit numbers
            in a counting system }
      Base = 2..16;

    { Nibble: unsigned 4-bit, 0 ... 2^4-1, half an 8-bit
byte}
      Nibble = 0..15;

    { Byte: unsigned 8-bit, 0 ... 2^8-1
      [Low(Byte)...High(Byte)] = [0 ... 255] }

    { Word: unsigned 16-bit, 0 ... 2^16-1
      [Low(Word)...High(Word)] = [0 ... 65535] }

    { ULongInt: unsigned 31-bit, 0 ... 2^31-1
      [0...High(LongInt)] = [0 ... 2147483647] }
      ULongInt = 0..2147483647;

  procedure WaitForExit(Msg: String);
  { Tells program to wait for keyboard input
  }
  procedure ClearDigits(var Data: array of Nibble);
  { Clear procedure assigns zero to each element in the array
  }
```

```
  function UpCaseStr(S: String): String;
  { UpCaseStr converts a string to uppercase
    for code page 'Windows 1251'
  }
  procedure CharacterTable(OutputFileName: String;
    FromCode, ToCode: Byte);
  { Output printable characters and codes in a text file
  }
  procedure ChangeNumberBase(InputNumber: String;
    CurrentBase, NewBase: Base; var OutputNumber: String);
  { Change the base of a counting system for a number
  }
  function PolynomOfIntegers(const Bs: Base;
    const Coefficients: array of Nibble): ULongInt;
  { Evaluates a uniform polynomial of one variable
    at value Bs. Horner scheme (algorithm).
    The coefficients are ordered in increasing powers of Bs:
    P(Bs) = Coefficients[0] + Coefficients[1]*Bs + ...
            + Coefficients[N]*(Bs**N)
  }
  procedure SHLWithRangeCheck(InpNumber, LowestValue,
    HighestValue: ULongInt; ShiftBy: Byte;
    var OutNumber: ULongInt; var Error: Boolean);
  { Shift the value of InpNumber to the left by ShiftBy bits
    with range check
  }

implementation
{ uses
  List of unit dependencies goes here... }
{ Implementation of procedures, and functions goes here... }

  procedure WaitForExit(Msg: String);
  { Write a message, wait for keyboard input,
    ends program execution
  }
  begin
    Writeln(Msg);
    Writeln('Press Enter for exit ...');
    Readln;
    Halt;
  end;

  procedure ClearDigits(var Data: array of Nibble);
  { Clear procedure assigns zero to each element in the array
  }
  var
    i: Byte;
  begin
    for i := 0 to High(Data) do
      Data[i] := 0;
```

```
end;

procedure CharacterTable(OutputFileName: String;
  FromCode, ToCode: Byte);
{ Output printable characters and codes in a text file
}
var
  OutputF: Text;
  i, Count: Integer;
begin
  Count := 0;
  Assign(OutputF, OutputFileName);
  Rewrite(OutputF);
  for i := FromCode to ToCode do
  begin
    Write(OutputF, Chr(i), ' (', i:3, ')');
    Count := Count + 1;
    if (Count Mod 8 <> 0) then
      Write(OutputF, #9) { #9 - Horizontal Tab }
    else
      Writeln(OutputF);
  end;
  Close(OutputF);
end;

function UpCaseStr(S: String): String;
{ UpCaseStr converts a string to uppercase
  for code page 'Windows 1251'
}
var
  i: Byte;
begin
  for i := 1 to Length(S) do
    case Ord(S[i]) of
      { English letters }
       97..122: S[i] := Chr(Ord(S[i])-32);
      { Russian letters }
           184: S[i] := Chr(168);
      224..255: S[i] := Chr(Ord(S[i])-32);
    end;
  UpCaseStr := S;
end;

function PolynomOfIntegers(const Bs: Base;
  const Coefficients: array of Nibble): ULongInt;
{ Evaluates a uniform polynomial of one variable
  at value Bs. Horner scheme (algorithm).
  The coefficients are ordered in increasing powers of Bs:
  P(Bs) = Coefficients[0] + Coefficients[1]*Bs + ...
          + Coefficients[N]*(Bs**N)
}
```

```
var
  i: Byte;
  DecimalValue: ULongInt;
begin
  DecimalValue := Coefficients[High(Coefficients)];
  for i := High(Coefficients)-1 DownTo Low(Coefficients) do
    DecimalValue := DecimalValue * Bs + Coefficients[I];
  PolynomOfIntegers := DecimalValue;
end;

function DigitOrdinalValue(const Digits: array of Char;
  Digit: Char): Integer;
var
  i, OrdinalValue: Integer;
begin
  OrdinalValue := -1;
  for i := 0 to High(Digits) do
    if (Digits[i] = Digit) then
      begin
        OrdinalValue := i;
        Break;
      end;
  DigitOrdinalValue := OrdinalValue;
end;

procedure GetDigitsDecimalValues(InputNumber: String;
  var DigitsCount: Byte;
  var DigitsDecimalValues: array of Nibble);
var
  i: Byte;
  Digit: Char;
  DigitDecimalValue: Nibble;
  Msg: String;
begin
  DigitsCount := 0;
  ClearDigits(DigitsDecimalValues);

  InputNumber := UpCaseStr(InputNumber);
  DigitsCount := Length(InputNumber);
  for i := 1 to DigitsCount do
  begin
    Digit := InputNumber[i];
    DigitDecimalValue:=DigitOrdinalValue(AllDigits, Digit);
    if (DigitDecimalValue <> -1) then
      DigitsDecimalValues[DigitsCount-i]:=DigitDecimalValue
    else
    begin
      Msg := 'Digit ' + Digit +
        ' is out of range binary - hexadecimal systems';
      WaitForExit(Msg);
    end;
```

```
  end;
end;

procedure SequenceOfNewDigits(NewBase: Base;
  Number: ULongInt; var Count: Byte;
  var NewBaseDigitsInDecimal: array of Nibble);
{ Output - digits of a number in new system

  Div: the value of x Div y is the value of x/y
  rounded in the direction of zero to the nearest integer.
  Result of Div - "quotient".
  Mod: the Mod operator returns the remainder
  obtained by dividing its operands.
}
var
  Quotient: ULongInt;
  Remainder: Nibble;
begin
  Count := 0;
  Quotient  := Number Div NewBase;
  Remainder  := Number Mod NewBase;
  while (Quotient <> 0) do
  begin
    NewBaseDigitsInDecimal[Count] := Remainder;
    Number := Quotient;
    Quotient  := Number Div NewBase;
    Remainder     := Number Mod NewBase;
    Count := Count + 1;
  end;
  NewBaseDigitsInDecimal[Count] := Remainder;

  { Fill with the symbol 0, indicating an absence of bits }
  if (NewBase=2) and (Count<7) then
    Count := 7;
end;

function DigitsToString(DigitsCount: Byte;
  NewBaseDigits: array of Nibble): String;
var
  i: Byte;
  CharDigit: String;
  NumberString: String;
  DigitOrdinalValue: Nibble;
begin
  NumberString := '';
  for i := 0 to DigitsCount do
  begin
    DigitOrdinalValue := NewBaseDigits[DigitsCount-i];
    CharDigit := AllDigits[DigitOrdinalValue];
    NumberString := NumberString + CharDigit;
  end;
```

```
    DigitsToString := NumberString;
  end;

  procedure ValidateBaseRange(Base: Integer);
  var
    Msg: String;
  begin
    if (Base < 2) or (Base > 16) then
      begin
        Msg := 'Base is out of range ' +
          'binary - hexadecimal systems';
        WaitForExit(Msg);
      end;
  end;

  procedure ChangeNumberBase(InputNumber: String;
    CurrentBase, NewBase: Base; var OutputNumber: String);
  { Change the base of a counting system for a number
  }
  var
    Sign: Char;
    Code: Integer;
    InputDecimalValue: ULongInt;
    DigitsCount, NewDigitsCount: Byte;
    NewBaseDigits, DigitsDecimalValues: array[Byte] of
Nibble;
  begin
    Sign := InputNumber[1];
    if Sign = '-' then
      WaitForExit('Input number have a value less than zero.
'
       + 'Try Abs(Input number).');

    ValidateBaseRange(CurrentBase);
    ValidateBaseRange(NewBase);

    if (CurrentBase = NewBase) then
    begin
      OutputNumber := InputNumber;
      Exit;
    end;

    DigitsCount := 0;
    NewDigitsCount := 0;
    ClearDigits(NewBaseDigits);
    ClearDigits(DigitsDecimalValues);

    if (CurrentBase = 10) then
      Val(InputNumber, InputDecimalValue, Code)
    else
    begin
```

```
      GetDigitsDecimalValues(InputNumber, DigitsCount,
        DigitsDecimalValues);
      InputDecimalValue := PolynomOfIntegers(CurrentBase,
        DigitsDecimalValues);
    end;
    SequenceOfNewDigits(NewBase, InputDecimalValue,
      NewDigitsCount, NewBaseDigits);
    OutputNumber := DigitsToString(NewDigitsCount,
      NewBaseDigits);
  end;

  procedure SHLWithRangeCheck(InpNumber, LowestValue,
    HighestValue: ULongInt; ShiftBy: Byte;
    var OutNumber: ULongInt; var Error: Boolean);
  { Shift the value of InpNumber to the left by ShiftBy bits
    with range check
  }
  begin
    {$R-}  { Disables the generation of range-checking code }
    OutNumber := InpNumber SHL ShiftBy;
    {$R+}  { Enables the generation of range-checking code }
    if (OutNumber >= LowestValue)
        and (OutNumber <= HighestValue) then
      Error := false
    else
      Error := true;
  end;

end.
```

Copyright – дословно, право делать копии

Право на охрану интеллектуальной собственности является конституционной нормой, и включает в себя также авторское право.

Что такое . . . авторское право?

Авторское право (англ. copyright) – исключительное право автора создавать копии и управлять оригиналом литературной, музыкальной или художественной работы.

Это право предоставляется автору и включает в себя право публикации, распространения и переработки оригинала; в случае смерти автора, право передается его наследникам на установленное законом количество лет.

Авторское право регулируется, в основном, гражданским законодательством и признается без какой-либо формальной регистрации.

Оповещение об авторском праве содержит:

– знак охраны авторского права в формах ©, (C) или Copyright;

– имя правообладателя;

– год первой публикации; если работа содержит части, опубликованные в разные годы, возможно указание лет публикаций через тире или запятую.

Например,

© инициалы фамилия, год публикации

или

© предприятие название, год публикации.