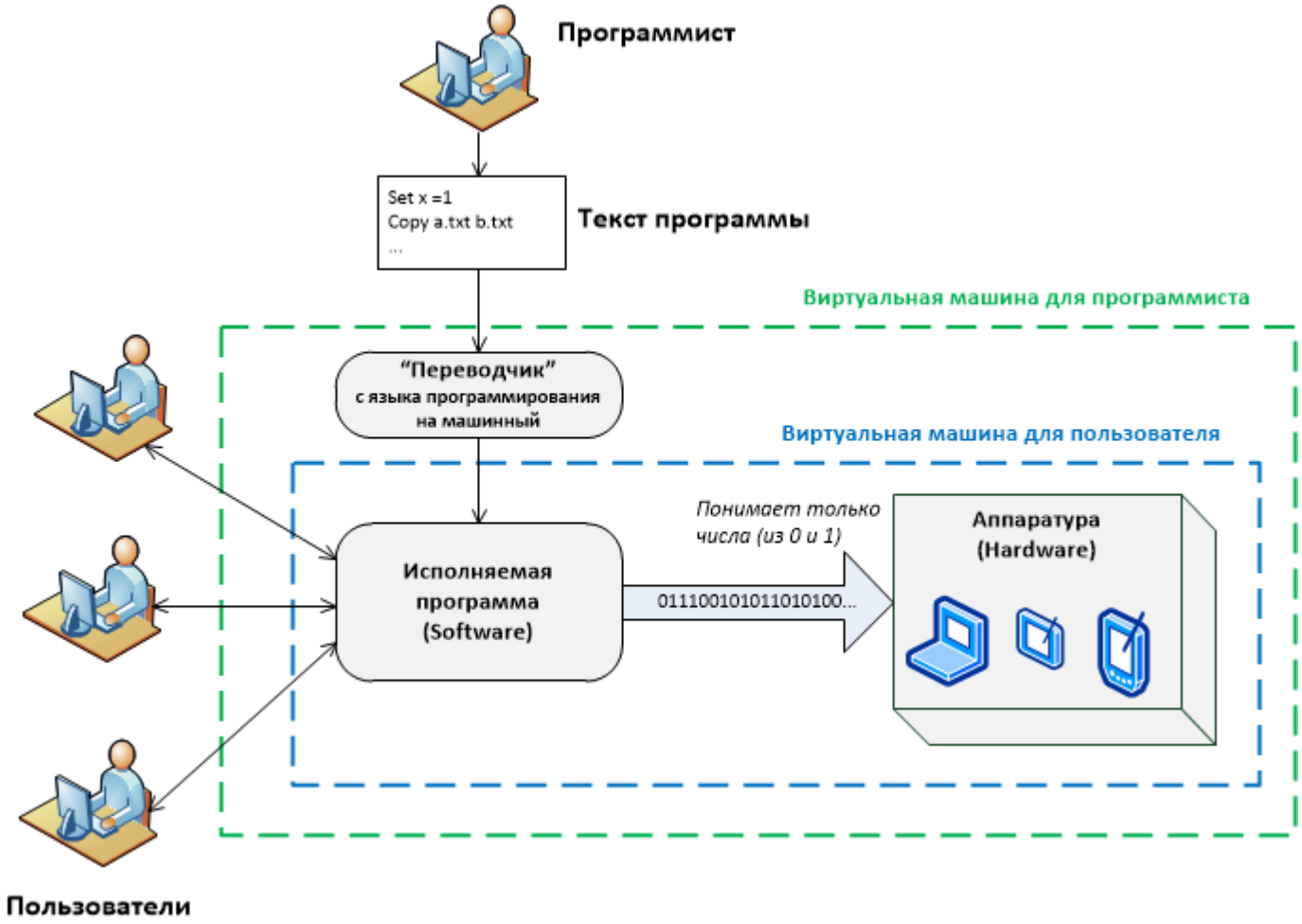
Лекция 2.

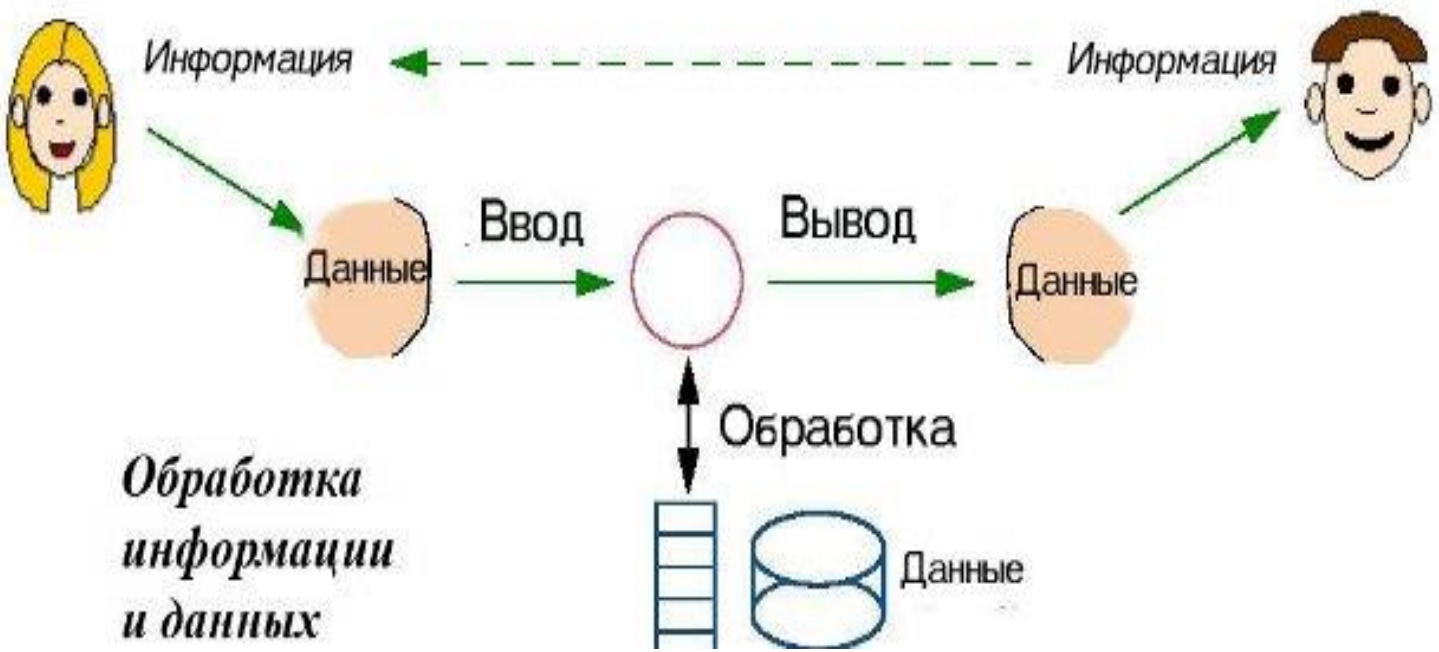
Архитектура фон Неймана. Числовые и символьные данные.



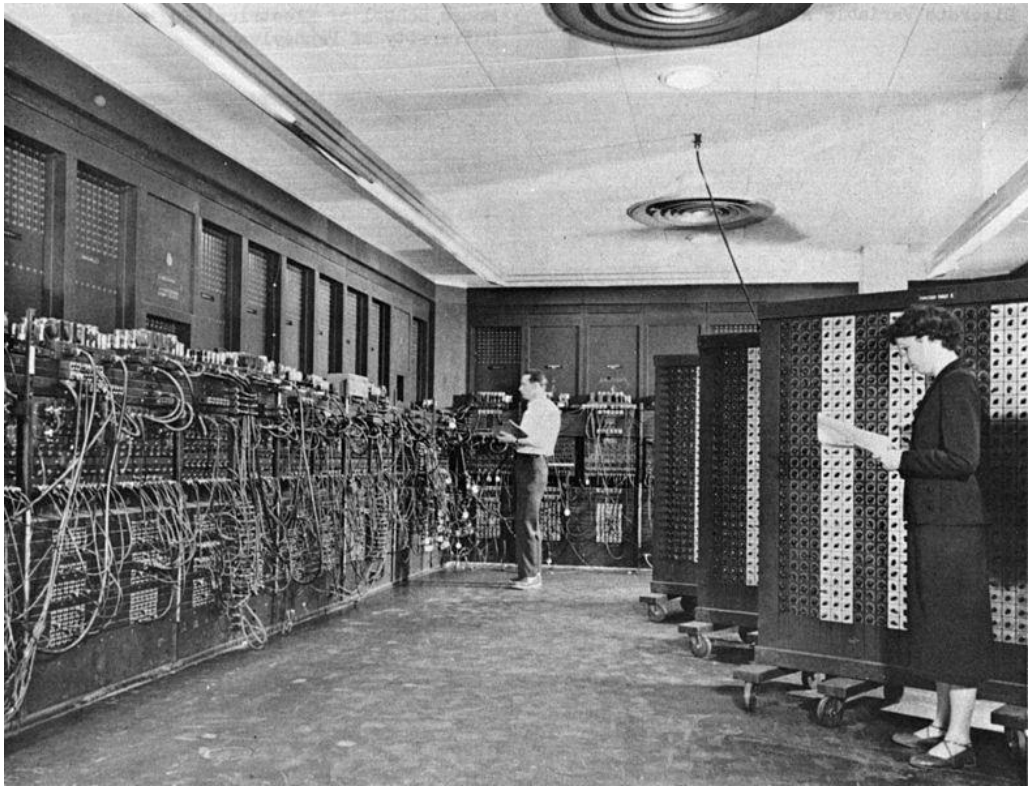
Программа на любом языке программирования выполняется компьютером в машинном коде.

Компьютеры могут: …

- Хранить и искать информацию в своей памяти.
- Выполнять (очень быстро) базовые операции над информацией: сравнение, замена, арифметические операции.
- Обмениваться данными с другими устройствами (компьютерами, мониторами, принтерами, телефонами, датчиками и т.д.).



Первые компьютерные системы: ограниченный набор исполняемых команд и программ



1945 год. Компьютер
ENIAC (Electronic Numerical Integrator and Computer)

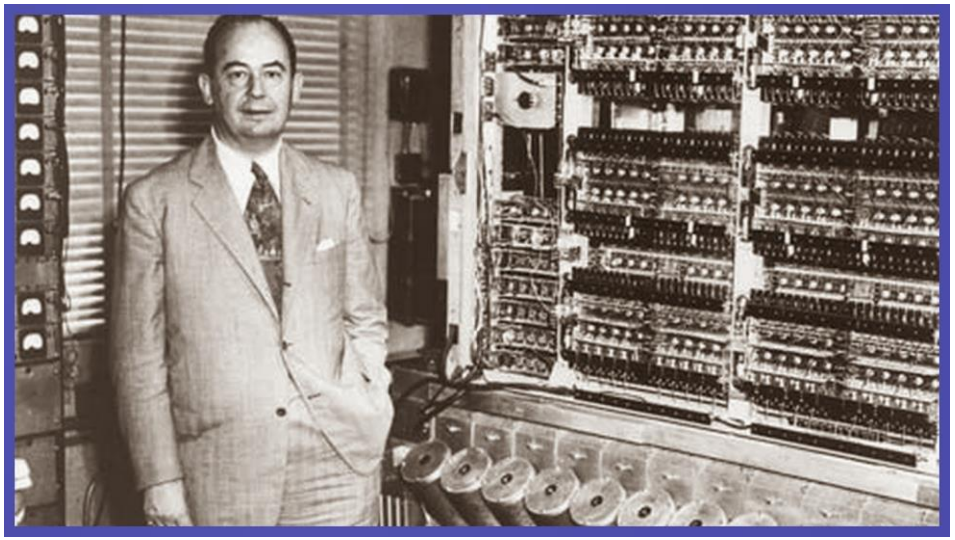
Площадь - 167 кв.м.
Вес - 27 тонн

Десятичная арифметика

Ввод/вывод данных – перфокарты

Программирование — ручная установка переключателей

Идея хранения программ в общей памяти



Джон фон Нейман
(1903-1957)

1946 год, статья Дж. фон Нейман, Г. Голдстайна и А. Беркса «Предварительное рассмотрение логической конструкции электронного вычислительного устройства» в журнале Nature

Принципы фон Неймана

- **Принцип двоичного кодирования.** Вся информация кодируется с помощью двоичных сигналов (двоичных цифр, битов) и разделяется на единицы, называемые словами.

Принципы фон Неймана

- **Принцип двоичного кодирования.** Вся информация кодируется с помощью двоичных сигналов (двоичных цифр, битов) и разделяется на единицы, называемые словами.
- **Принцип однородности памяти.** Программы и данные хранятся в одной и той же памяти. Компьютер не различает, что хранится в данной ячейке памяти — число, текст или команда.

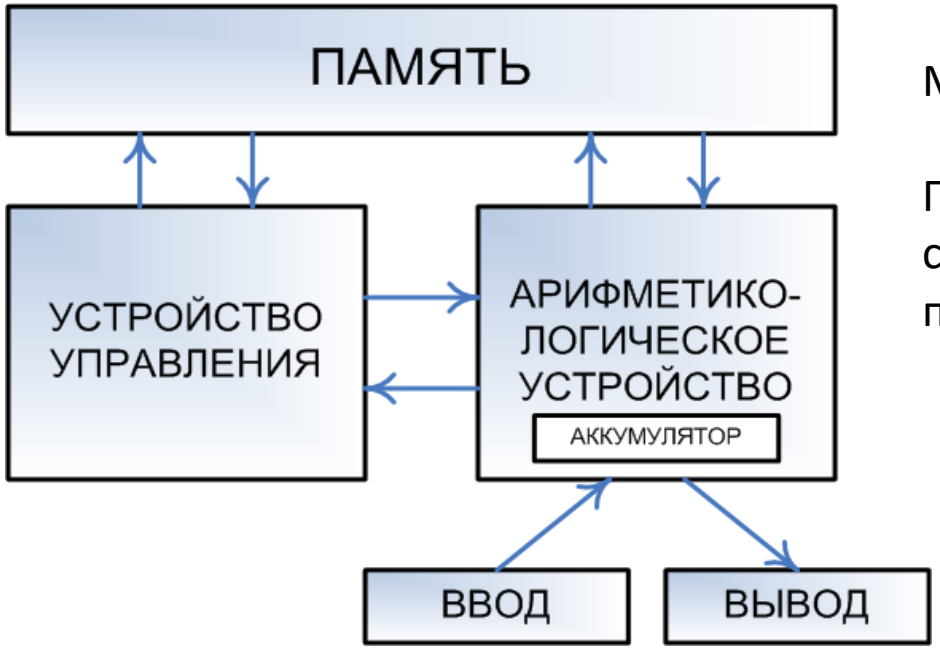
Принципы фон Неймана

- **Принцип двоичного кодирования.** Вся информация кодируется с помощью двоичных сигналов (двоичных цифр, битов) и разделяется на единицы, называемые словами.
- **Принцип однородности памяти.** Программы и данные хранятся в одной и той же памяти. Компьютер не различает, что хранится в данной ячейке памяти — число, текст или команда.
- **Принцип адресуемости памяти.** Память состоит из пронумерованных ячеек; процессору в произвольный момент времени доступна любая ячейка. Можно давать имена областям памяти.

Принципы фон Неймана

- **Принцип двоичного кодирования.** Вся информация кодируется с помощью двоичных сигналов (двоичных цифр, битов) и разделяется на единицы, называемые словами.
- **Принцип однородности памяти.** Программы и данные хранятся в одной и той же памяти. Компьютер не различает, что хранится в данной ячейке памяти — число, текст или команда.
- **Принцип адресуемости памяти.** Память состоит из пронумерованных ячеек; процессору в произвольный момент времени доступна любая ячейка. Можно давать имена областям памяти.
- **Принцип последовательного программного управления.** Предполагает, что программа состоит из набора команд, которые выполняются процессором автоматически друг за другом в определенной последовательности.

Архитектура фон Неймана



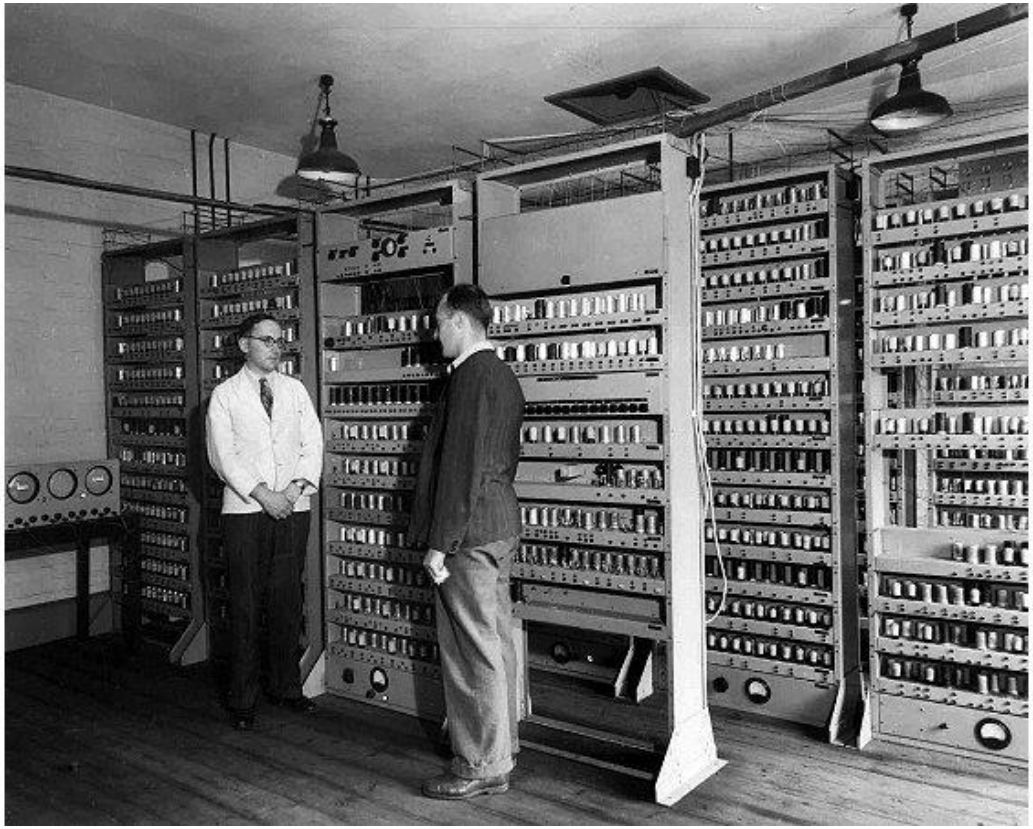
Машина работает в двоичном коде.

Программы и данные хранятся совместно в одном и том же пространстве памяти.

Программа для фон-неймановской машины – последовательность нулей и единиц, одни участки которой кодируют команды, а другие – данные.

Подобная последовательность называется **машинным кодом**.

Первый компьютер с архитектурой фон Неймана



1949 год. Компьютер EDSAC
(Electronic Delay Storage Automatic Computer)
Кембриджский университет

Поддерживал язык ассемблера и подпрограммы.
В 1952 году на нем была написана первая игра «Крестики-нолики».

Основы двоичной системы

1 бит = 0, 1
1 байт = 8 бит
1 слово = 8 байт = 64 бит (раньше 4 байт = 32 бит). Равно разрядности регистров процессора и/или шины данных.

Основы двоичной системы

1 бит = 0, 1
1 байт = 8 бит
1 слово = 8 байт = 64 бит (раньше 4 байт = 32 бит). Равно разрядности регистров процессора и/или шины данных.

Сколько разных чисел можно закодировать 1 байтом?

Основы двоичной системы

1 бит = 0, 1
1 байт = 8 бит
1 слово = 8 байт = 64 бит (раньше 4 байт = 32 бит). Равно разрядности регистров процессора и/или шины данных.

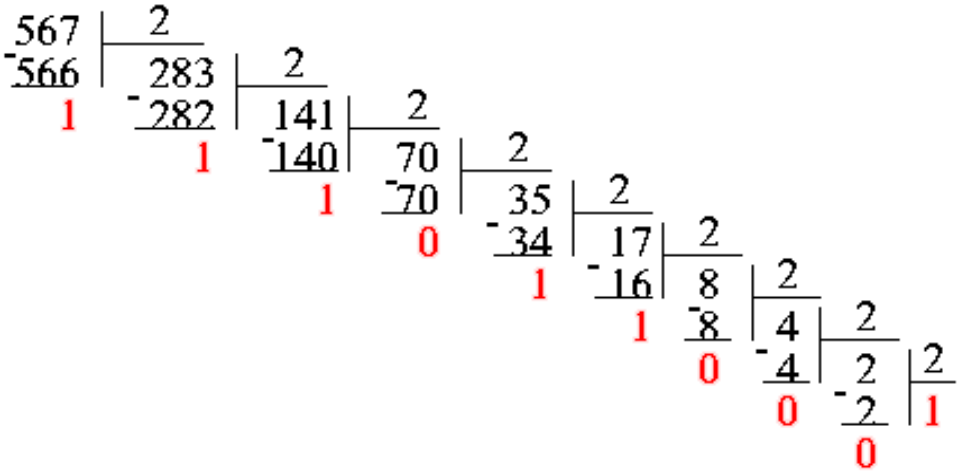
Сколько разных чисел можно закодировать 1 байтом?

$00000000_2 = 0_{10}$
$00000001_2 = 1_{10}$
$00000010_2 = 2_{10}$
$00000011_2 = 3_{10}$
. . .
$11111111_2 = 255_{10}$

Целые положительные числа

Перевод из десятичной в двоичную:

1. Если текущее число меньше 2, то записать его в младший разряд результата, выполнение прекратить. Иначе, разделить текущее число с остатком на 2.
2. Остаток записать в младший разряд результата.
3. Применить пункт 1 к частному.



$$567_{10} = 1000110111_2$$

Перевод из двоичной в десятичную:

$1\cdot2^9+0\cdot2^8+0\cdot2^7+0\cdot2^6+1\cdot2^5+1\cdot2^4+0\cdot2^3+1\cdot2^2+1\cdot2^1+1\cdot2^0=512+0+0+0+32+16+0+4+2+1 =$ **567**

Целые числа со знаком

Как учесть знак числа, как хранить отрицательные числа?

Целые числа со знаком

Как учесть знак числа, как хранить отрицательные числа?

Под знак числа отводят один бит (двоичный разряд). Остальные разряды определяют модуль числа.

Целые числа со знаком

Как учесть знак числа, как хранить отрицательные числа?

Под знак числа отводят один бит (двоичный разряд). Остальные разряды определяют модуль числа.

Прямой код

Старший бит – знак числа (0, если «+» и 1, если «-»). Остальные биты не меняются. Два недостатка:

$+0 = 0.0000000_2$ и $-0 = 1.0000000_2$

$a = 65_{10} = 0.1000001_2$

$-a = -65_{10} = 1.1000001_2$

$a+(-a) = 0.0110010+1.0110010 = 1.1100100 \neq 0$

Целые числа со знаком

Как учесть знак числа, как хранить отрицательные числа?

Под знак числа отводят один бит (двоичный разряд). Остальные разряды определяют модуль числа.

Прямой код

Старший бит – знак числа (0, если «+» и 1, если «-»). Остальные биты не меняются. Два недостатка:

$+0 = 0.0000000_2$ и $-0 = 1.0000000_2$

$a = 65_{10} = 0.1000001_2$

$-a = -65_{10} = 1.1000001_2$

$a+(-a) = 0.0110010+1.0110010 = 1.1100100 \neq 0$

Обратный код

Старший бит – знак числа. Остальные биты инвертируются (0 -> 1, 1 -> 0).

$+0 = 0.0000000_2$ и $-0 = 1.1111111_2$

$a=65_{10}=0.1000001_2$

$-a=-65_{10}=1.0111110_2$

В сложении участвуют все биты, единица переноса из знакового разряда прибавляется к младшему разряду суммы.

$a+(-a) = 0.1000001_2+1.0111110_2 = 1.1111111_2 = -0$

Целые числа со знаком

Дополнительный код

Старший бит – знак числа. Остальные биты инвертируются, к результату прибавляется 1.

Единственная запись нуля: 0 = 0.0000000

В сложении участвуют все биты, единица переноса из знакового разряда отбрасывается.

$a=65_{10}=0.1000001_2$

$-a=-65_{10}=1.0111111_2$

$a+(-a) = 0.1000001_2+1.0111111_2 = 0.0000000_2 = 0$ (*1 в 9 разряде отбросили*)

Целые числа со знаком

Дополнительный код

Старший бит – знак числа. Остальные биты инвертируются, к результату прибавляется 1.

Единственная запись нуля: 0 = 0.0000000

В сложении участвуют все биты, единица переноса из знакового разряда отбрасывается.

$a=65_{10}=0.1000001_2$

$-a=-65_{10}=1.0111111_2$

$a+(-a) = 0.1000001_2+1.0111111_2 = 0.0000000_2 = 0$ (*1 в 9 разряде отбросили*)

Пример: **145 – 102 = 43**

$$
\begin{array}{r}
[x_1]_o=0.10010001 \\
+[x_2]_o=1.10011001 \\
\hline
10.00101010 \\
\longrightarrow +1 \\
\hline
0{,}00101011
\end{array}
$$

Обратный код

$$
\begin{array}{r}
[x_1]_д=0.10010001 \\
+[x_2]_д=1.10011010 \\
\hline
10.00101011 \\
\hline
0{,}00101011
\end{array}
$$

Дополнительный код

Вещественные числа

1985 год, стандарт IEEE754 «Двоичная арифметика с плавающей точкой»

Число одинарной точности от $-3{,}4 \cdot 10^{38}$ до $3{,}4 \cdot 10^{38}$ хранится в 32-битовом контейнере, разделенном на три части:

- S (1 бит) для хранения знака
- E (8 бит) для хранения экспоненты (порядка)
- M (23 бита) для хранения мантиссы

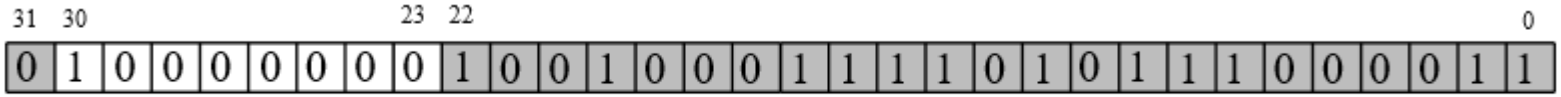
Внутренности числа с плавающей запятой.

S	EXPONENT	MANTISSA

Три части числа с плавающей запятой.

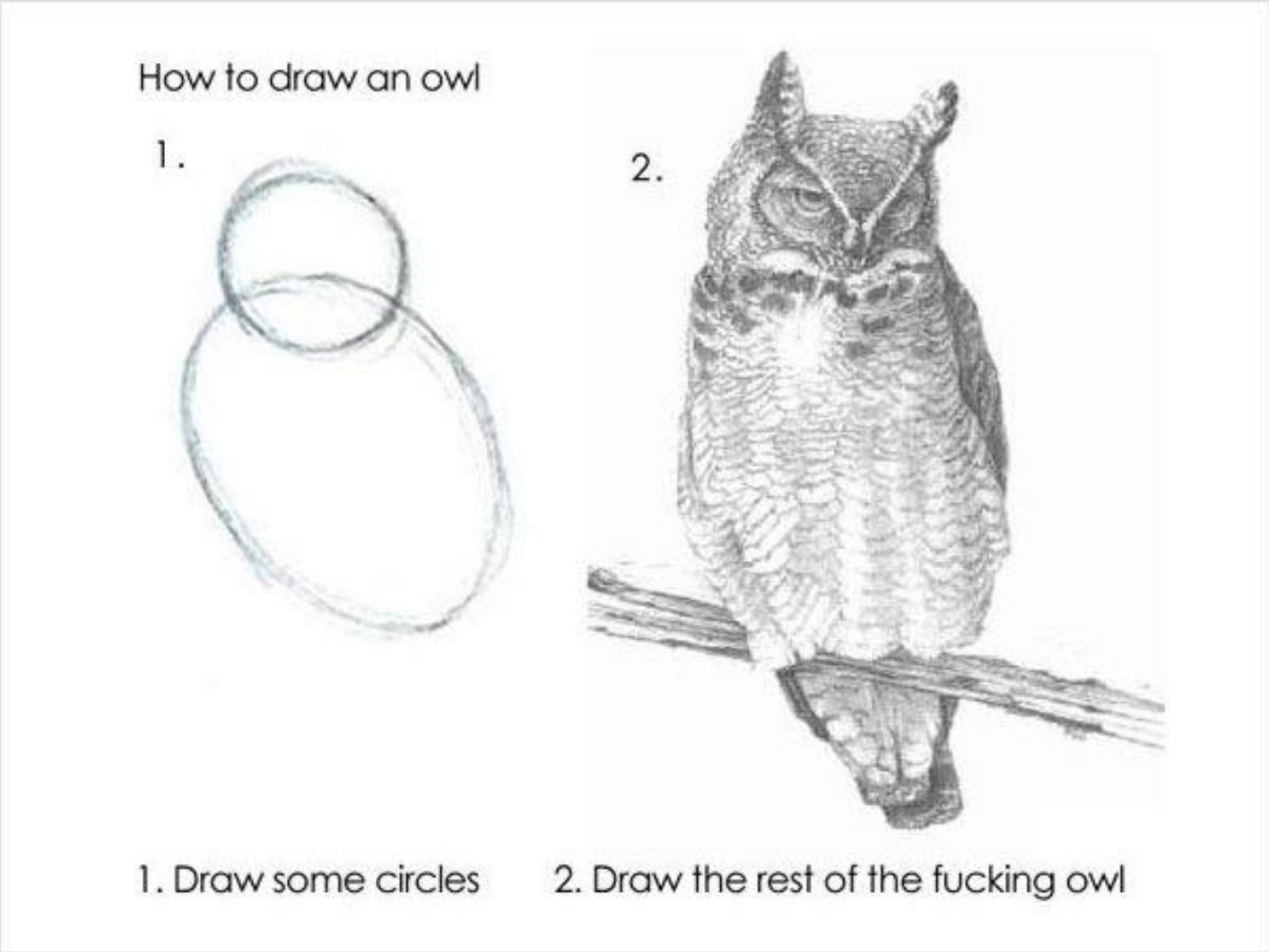
Числа представляются с помощью формулы $(-1)^S * 1,M * 2^{(E-127)}$

$$3,14 = (-1)^0 * 1,57 * 2^{(128-127)}$$



Двоичное представление с плавающей точной числа 3,14.

Вещественные числа



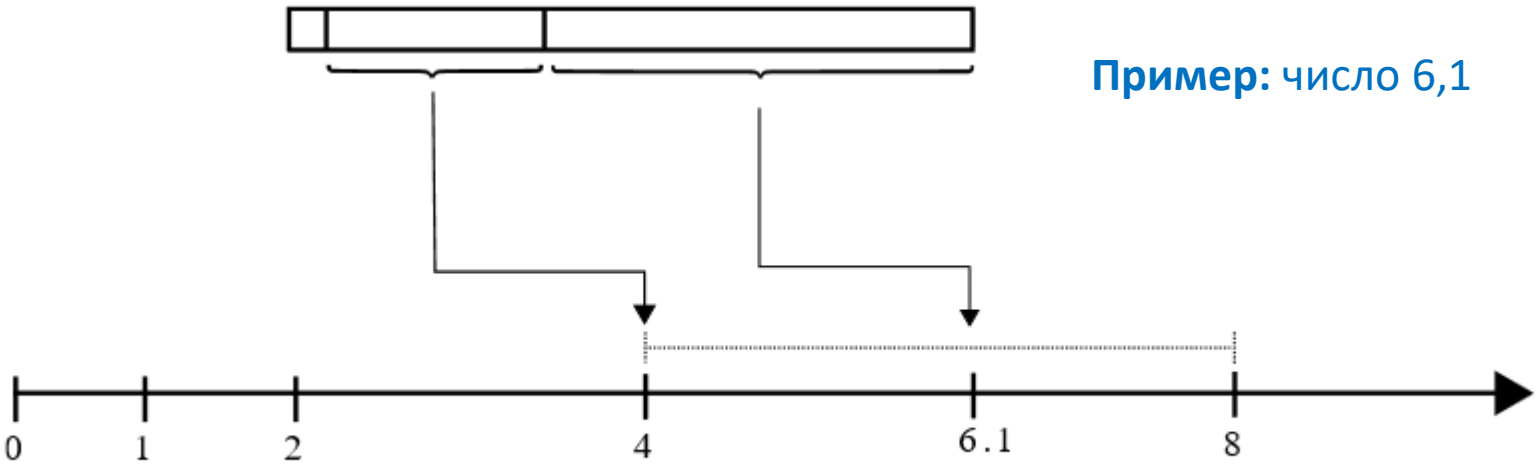
Вещественные числа

«Наглядное объяснение чисел с плавающей запятой» https://habrahabr.ru/post/337260/

S	WINDOW	OFFSET

Три части числа с плавающей запятой.

- **Окно** – число, определяющее интервал [0, 1], [1, 2], [2, 4], [4,8], …, [2^{127}, 2^{128}], в который попадает целая часть числа.
- Отрезок Window делится на 2^{23} = 8 388 608 сегментов.
- **Смещение** определяет номер сегмента, в которые попадает число.



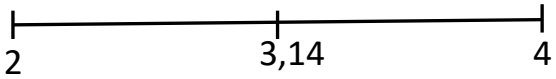
Пример: число 6,1

Значение 6,1 аппроксимированное с помощью числа с плавающей запятой.

Вещественные числа

Число 3,14

- 3,14 > 0 , поэтому **S=0**.
- 2 < 3,14 < 4, поэтому **E=128** (так как окно – это 2^{E-127}).
- (3,14 – 2)/(4-2) = 0,57 – местоположение точки 3,14 на интервале [2, 4].



- Вычисляем номер смещения: M = 2^{23} * 0,57 = 4781507.

В двоичном виде:

- S = 0 = 0b
- E = 128 = 10000000b
- M = 4781507 = 10010001111010111000011b

31	30							23	22																						0
0	1	0	0	0	0	0	0	0	1	0	0	1	0	0	0	1	1	1	1	0	1	0	1	1	1	0	0	0	0	1	1

Двоичное представление с плавающей точной числа 3,14.

Шестнадцатеричная система

Для представления двоичных данных в удобном виде.

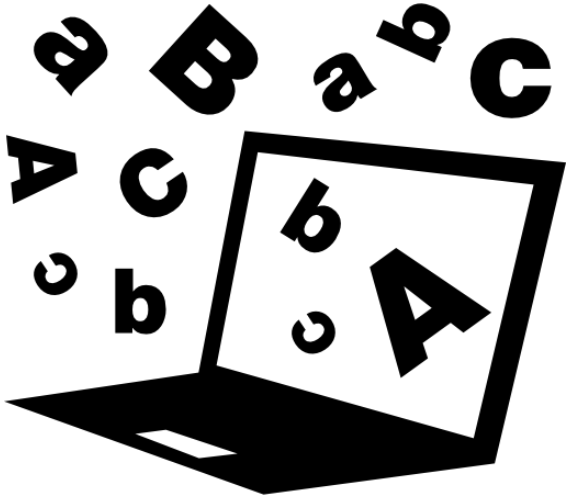
Двоичная тетрада = один 16-ичный знак:

$0000_2=0_{16}$
$0001_2=1_{16}$
$0010_2=2_{16}$
$0011_2=3_{16}$
$0100_2=4_{16}$
$0101_2=5_{16}$
$0110_2=6_{16}$
$0111_2=7_{16}$
$1000_2=8_{16}$
$1001_2=9_{16}$
$1010_2=A_{16}$
$1011_2=B_{16}$
$1100_2=C_{16}$
$1101_2=D_{16}$
$1110_2=E_{16}$
$1111_2=F_{16}$

Символы, буквы, знаки

Для работы с символами в компьютере нужно:

1. Определить множество символов, которые должны отображаться.
2. Сопоставить каждому символу число (код).
3. Определить способ хранения этих чисел в памяти компьютера (кодирование).



Первые кодировки

Вначале требовалось кодировать минимум символов:

- 10 цифр
- 26 заглавных букв английского алфавита
- 26 строчных букв
- математические знаки +-/*=><%
- знаки препинания .,!?:;’”
- различные скобки (){}[]
- служебные символы _^%$@|
- 32 непечатных управляющих символов

Первые кодировки

Вначале требовалось кодировать минимум символов:

- 10 цифр
- 26 заглавных букв английского алфавита
- 26 строчных букв
- математические знаки +-/*=><%
- знаки препинания .,!?:;’”
- различные скобки (){}[]
- служебные символы _^%$@|
- 32 непечатных управляющих символов

Для этого хватало **128 символов**.

Для хранения информации использовались **7 бит** в байте (старший бит – для контроля или других целей).

Кодовая таблица ASCII (American Standard Code for Information Interchange)

символ	*10-ti код*	*2-ti код*	*символ*	*10-ti код*	*2-ti код*	*символ*	*10-ti код*	*2-ti код*	*символ*	*10-ti код*	*2-ti код*
	32	00100000	8	56	00111000	P	80	01010000	h	104	01101000
!	33	00100001	9	57	00111001	Q	81	01010001	i	105	01101001
"	34	00100010	:	58	00111010	R	82	01010010	j	106	01101010
#	35	00100011	;	59	00111011	S	83	01010011	k	107	01101011
$	36	00100100	<	60	00111100	T	84	01010100	l	108	01101100
%	37	00100101	=	61	00111101	U	85	01010101	m	109	01101101
&	38	00100110	>	62	00111110	V	86	01010110	n	110	01101110
'	39	00100111	?	63	00111111	W	87	01010111	o	111	01101111
(	40	00101000	@	64	01000000	X	88	01011000	p	112	01110000
)	41	00101001	A	65	01000001	Y	89	01011001	q	113	01110001
*	42	00101010	B	66	01000010	Z	90	01011010	r	114	01110010
+	43	00101011	C	67	01000011	[	91	01011011	s	115	01110011
,	44	00101100	D	68	01000100	\	92	01011100	t	116	01110100
-	45	00101101	E	69	01000101	]	93	01011101	u	117	01110101
.	46	00101110	F	70	01000110	^	94	01011110	v	118	01110110
/	47	00101111	G	71	01000111	_	95	01011111	w	119	01110111
0	48	00110000	H	72	01001000	`	96	01100000	x	120	01111000
1	49	00110001	I	73	01001001	a	97	01100001	y	121	01111001
2	50	00110010	J	74	01001010	b	98	01100010	z	122	01111010
3	51	00110011	K	75	01001011	c	99	01100011	{	123	01111011
4	52	00110100	L	76	01001100	d	100	01100100	|	124	01111100
5	53	00110101	M	77	01001101	e	101	01100101	}	125	01111101
6	54	00110110	N	78	01001110	f	102	01100110	~	126	01111110
7	55	00110111	O	79	01001111	g	103	01100111	□	127	01111111

Кодовая таблица ASCII (American Standard Code for Information Interchange)

символ	10-ti код	2-ti код	символ	10-ti код	2-ti код	символ	10-ti код	2-ti код	символ	10-ti код	2-ti код
	32	00100000	8	56	00111000	P	80	01010000	h	104	01101000
!	33	00100001	9	57	00111001	Q	81	01010001	i	105	01101001
"	34	00100010	:	58	00111010	R	82	01010010	j	106	01101010
#	35	00100011	;	59	00111011	S	83	01010011	k	107	01101011
$	36	00100100	<	60	00111100	T	84	01010100	l	108	01101100
%	37	00100101	=	61	00111101	U	85	01010101	m	109	01101101
&	38	00100110	>	62	00111110	V	86	01010110	n	110	01101110
'	39	00100111	?	63	00111111	W	87	01010111	o	111	01101111
(	40	00101000	@	64	01000000	X	88	01011000	p	112	01110000
)	41	00101001	A	65	01000001	Y	89	01011001	q	113	01110001
*	42	00101010	B	66	01000010	Z	90	01011010	r	114	01110010
+	43	00101011	C	67	01000011	[	91	01011011	s	115	01110011
,	44	00101100	D	68	01000100	\	92	01011100	t	116	01110100
-	45	00101101	E	69	01000101	]	93	01011101	u	117	01110101
.	46	00101110	F	70	01000110	^	94	01011110	v	118	01110110
/	47	00101111	G	71	01000111	_	95	01011111	w	119	01110111
0	48	00110000	H	72	01001000	`	96	01100000	x	120	01111000
1	49	00110001	I	73	01001001	a	97	01100001	y	121	01111001
2	50	00110010	J	74	01001010	b	98	01100010	z	122	01111010
3	51	00110011	K	75	01001011	c	99	01100011	{	123	01111011
4	52	00110100	L	76	01001100	d	100	01100100	|	124	01111100
5	53	00110101	M	77	01001101	e	101	01100101	}	125	01111101
6	54	00110110	N	78	01001110	f	102	01100110	~	126	01111110
7	55	00110111	O	79	01001111	g	103	01100111	□	127	01111111

Почему большие и маленькие буквы идут не сразу друг за другом?

Кодовые страницы

Как кодировать символы других алфавитов?

Кодовые страницы

Как кодировать символы других алфавитов?

Нужно использовать все 8 бит в байте. Числам от 0 до 255 сопоставляются графические образы символов – появляются разные кодовые страницы.

Шрифты-кодировки

CP866 (DOS)

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

CP1251 (Windows)

Ѓ 128	ѓ 129	‚ 130	ℓ 131	„ 132	… 133	† 134	‡ 135	€ 136	‰ 137	Ѣ 138	‹ 139	ѣ 140	Ѝ 141	Ѳ 142	Ѵ 143
ѓ 144	‘ 145	’ 146	“ 147	” 148	• 149	– 150	— 151	̃ 152	™ 153	ѣ 154	› 155	ѳ 156	ѝ 157	ѳ 158	ѵ 159
nbsp 160	Ў 161	Ы 162	Ѓ 163	¤ 164	ы 165	¦ 166	§ 167	Ё 168	© 169	Ю 170	« 171	¬ 172	shy 173	® 174	Я 175
° 176	± 177	ы 178	ѓ 179	’ 180	µ 181	¶ 182	· 183	ё 184	№ 185	ю 186	» 187	ѣ 188	ю 189	я 190	я 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

Кодовые страницы

КОИ-8 (код обмена информацией) – восьмибитовая кодовая страница (исторически первая для кириллических символов).

Пришла из Unix, широко применялась в электронной почте.

Таблица 1.3. Кодировка КОИ-8

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

Кодовые страницы

КОИ-8 (код обмена информацией) – восьмибитовая кодовая страница (исторически первая для кириллических символов).

Пришла из Unix, широко применялась в электронной почте.

Таблица 1.3. Кодировка КОИ-8

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

Задача:

1. Записать в КОИ-8 свое имя и фамилию. Прочитать строку с помощью 7-битного “приложения”.

Символьные строки

C-строки (ASCIIZ-строки)

Строка = массив символов, концом строки считается символ с ASCII-кодом 0.

“abc” = [97 98 99 0]

Pascal-строки

Строка = массив символов, длина определяется счетчиком в первом элементе этого массива (1 байт, максимальная длина строки – 255 символов).

“abc” = [3 97 98 99]

Недостатки однобайтовых кодировок

- Одновременно работать можно лишь с 256 символами (первые 128 зарезервированы).

Недостатки однобайтовых кодировок

- Одновременно работать можно лишь с 256 символами (первые 128 зарезервированы).
- Шрифты привязаны к конкретной кодировке.

Недостатки однобайтовых кодировок

- Одновременно работать можно лишь с 256 символами (первые 128 зарезервированы).
- Шрифты привязаны к конкретной кодировке.
- Каждая кодировка представляет свой набор символов и конвертация из одной в другую возможна только с частичными потерями (отсутствующие символы заменяются на графически похожие).

Недостатки однобайтовых кодировок

- Одновременно работать можно лишь с 256 символами (первые 128 зарезервированы).
- Шрифты привязаны к конкретной кодировке.
- Каждая кодировка представляет свой набор символов и конвертация из одной в другую возможна только с частичными потерями (отсутствующие символы заменяются на графически похожие).
- Перенос файлов между устройствами под управлением разных операционных систем затруднителен. Нужно либо иметь программу-конвертер, либо таскать вместе с файлом дополнительные шрифты.

Недостатки однобайтовых кодировок

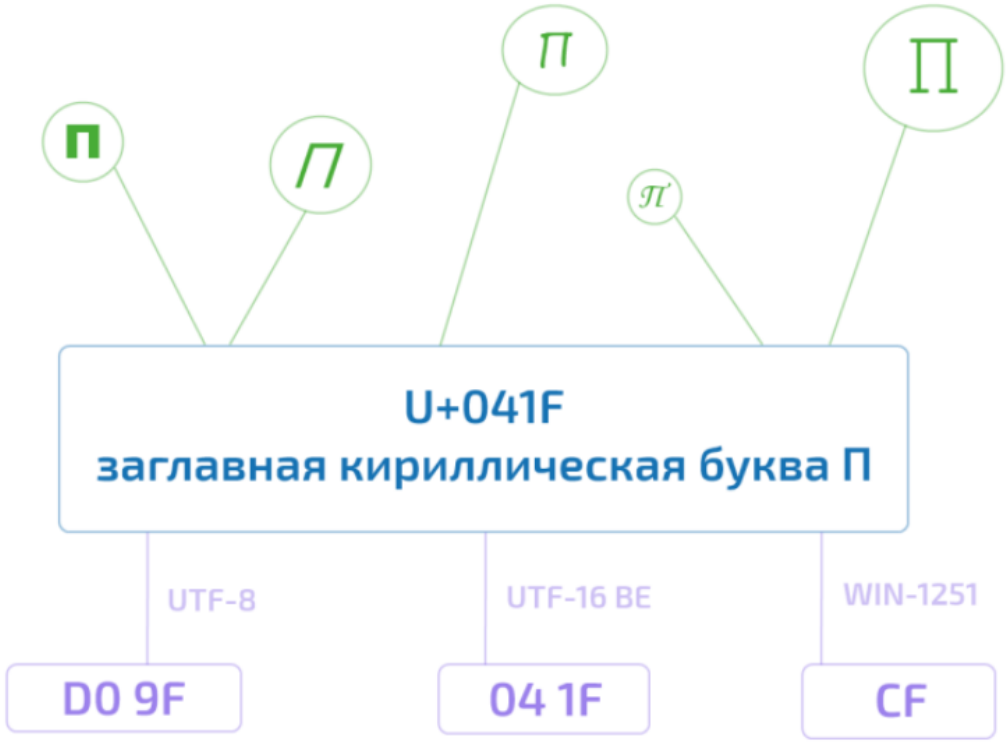
- Одновременно работать можно лишь с 256 символами (первые 128 зарезервированы).
- Шрифты привязаны к конкретной кодировке.
- Каждая кодировка представляет свой набор символов и конвертация из одной в другую возможна только с частичными потерями (отсутствующие символы заменяются на графически похожие).
- Перенос файлов между устройствами под управлением разных операционных систем затруднителен. Нужно либо иметь программу-конвертер, либо таскать вместе с файлом дополнительные шрифты.
- В мире существуют неалфавитные системы письма (иероглифическая письменность), которые в однобайтной кодировке непредставимы в принципе.

Unicode

Чёткое разграничение между символами, их представлением в компьютере и их отображением на устройстве вывода.

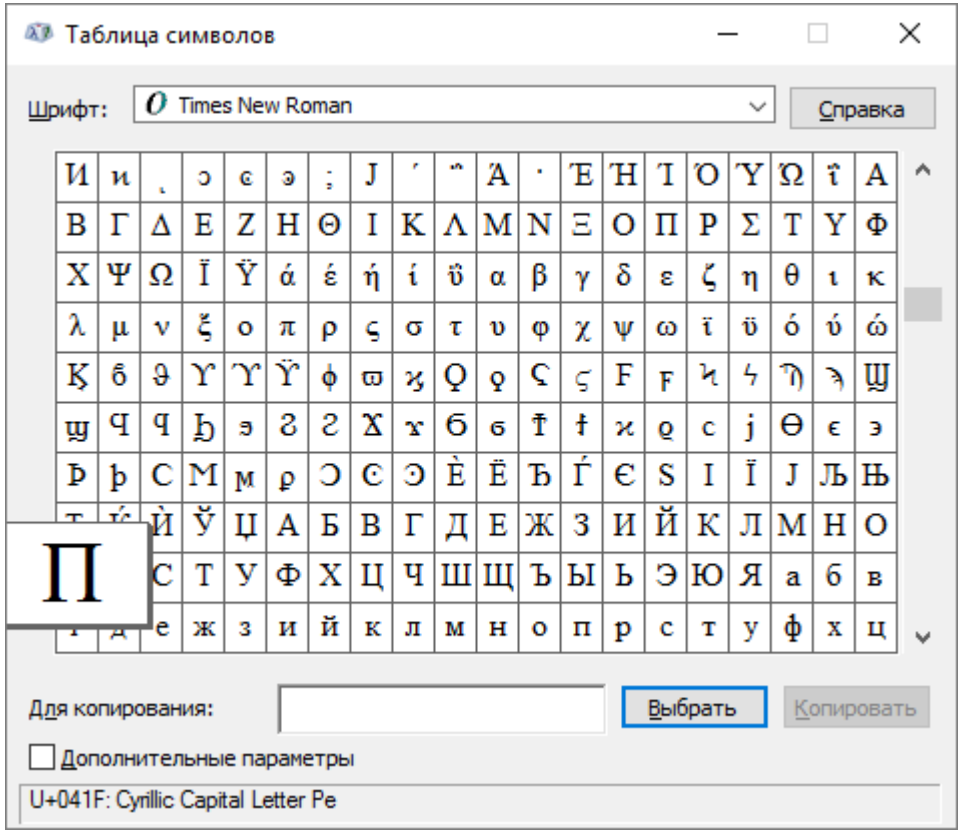
1. В стандарте определяются абстрактные Unicode-символы.

- Каждому символу сопоставляется целое число (кодовая позиция, code point). Зарезервировано 1 114 112 кодовых позиций.



Unicode

- Сайт http://www.unicode.org/charts/
- Утилита charmap



Unicode

Hello -> U+0048 U+0065 U+006C U+006C U+006F

Как хранить в памяти компьютера?

Unicode

Hello -> U+0048 U+0065 U+006C U+006C U+006F

Как хранить в памяти компьютера?

1. Два байта на каждый символ (65535 символов).

UCS-2 (Universal Character Set, два байта на символ).

Hello -> 00 48 00 65 00 6C 00 6C 00 6F

Hello -> 48 00 65 00 6C 00 6C 00 6F 00

Unicode

Hello -> U+0048 U+0065 U+006C U+006C U+006F

Как хранить в памяти компьютера?

1. Два байта на каждый символ (65535 символов).

UCS-2 (Universal Character Set, два байта на символ).

Hello -> 00 48 00 65 00 6C 00 6C 00 6F

Hello -> 48 00 65 00 6C 00 6C 00 6F 00

2. Мультибайтовая кодировка UTF-8

Unicode Transformation Format, 8 бит

1 байт – 0xxxxxxx

2 байта – 110xxxxx 10xxxxxx

3 байта – 1110xxxx 10xxxxxx 10xxxxxx

4 байта – 11110xxx 10xxxxxx 10xxxxxx 10xxxxxx

Представление первых 127 символов в Unicode совпадает с представлением символов ASCII.

Hello -> 48 65 6C 6C 6F