Представяне на символна и текстова информация

В компютърните системи информацията се съхранява само в двоичен код, използваните символи при съставяне на даден текст трябва да се представят като комбинация от двоични цифри (1 и 0). При натискане на клавиш от клавиатурата, който съответства на даден символ, се формира определен код (scan cod) – двоично число, определящо се от поредния номер на клавиша. Номерът на натиснатия клавиш обикновено не е свързан със съответния символ. Връзката между формирания 'scan cod' и символа се определя от специална таблица. В нея на всеки символ, използван от компютърната система, се присвоява двоично число. Когато се записва текст в паметта на компютъра, всъщност се записва последователност от двоични числа, съответстващи на символите от текста. Те не се различават по начина на записване от обикновените цели числа. Разглеждането на тази информация като текст е условно, тъй като трябва да има предварително уточнение, че това е текст, а не обикновени числа. За целта е създадена специална кодова таблица описваща връзката между изображението и кода на символа. Тази кодова таблица се нарича ASCII (American Standard Code for Information Interchange). ASCII представлява таблица от печатни символи и някой специални управляващи символи. На всеки символ отговаря уникален код в диапазона от 0 до 255. Сама по себе си ASCII представлява код за представяне на десетични цифри, символи на латиница и кирилица, препинателни знаци и управляващи символи (виж. Таблица 1).

В Таблица 1 е даден пълният набор от ASCII символи. В таблица 2 са дадени управляващите ASCII символи и тяхното предназначение. Символите от тази таблица изпълняват някой действия и те се извеждат в съчетание с клавиш **Ctrl**. В таблица 1 са дадени обозначенията на управляващите ASCII символи, но не и на техните изображения при печат. Останалите символи от 32 до 254 не са управляващи и се визуализират както са показани в таблицата.

До тук всичко е добре, но съществуват множество езици, които при съвременното развитие и използване на компютърните системи изискват много повече от старата ASCII таблица и пригаждането ѝ към различни азбуки. Особено големи са изискванията на литературата – тя използва много характерни особености на различните азбуки като ударения, транскрипции и други.

Решението на този въпрос е въвеждането на системата за кодиране Unicode. Unicode съдържа в себе си много повече кодове. Това се дължи на това че един символ в Unicode вече се кодира с 2 или повече байта. Това позволява таблицата да съдържа най-малко 65536 символа. Това позволява да бъдат кодирани символите на много езици, както и други символи, използвани в математиката и инженерните науки. Всеки един набор символи може без загуба да бъде преобразуван към Unicode.

Най-разпространените Unicode кодирания се означават с UTF-n (Unicode Transformation Format), където число n определя броя битове в основната единица, използвана от кодирането.

Две много разпространени кодирания са UTF-16 и UTF-8. При UTF-16, което се използва от съвременните версии на Microsoft Windows системите, всеки символ е представен от една или две 16-битови (двубайтови) думи. Unix-подобните операционни системи, включително и Linux използват друга схема на кодиране, наречена UTF-8, където всеки Unicode символ е представен като един или повече байтове (общо до четири; по-стара версия на стандарта разрешава до шест).

UTF-8 има няколко интересни свойства, които го правят подходящ за тази цел. Първо, ASCII символите се кодират по един и същ начин при ASCII и при UTF-8. Това значи, че всеки един ASCII текстови файл също така е и коректно кодиран UTF-8 Unicode текстови файл, представляващ същият текст. В допълнение когато се кодират в UTF-8 символи, заемащи повече от един байт, символите от ASCII набора никога не се използват. Това гарантира между другите неща, че ако софтуерен код обработва подобен файл като чист ASCII, не-ASCII символите се игнорират или в най-лошият случай се третират като случаен боклук, но те не могат да бъдат прочетени като ASCII символи (което може случайно да доведе до формално коректен, но потенциално проблемни конфигурационни опции в конфигурационен файл или да доведе до други непредвидени резултати). Имайки предвид важността на текстовите файлове в Unix тези характеристики за важни. Благодарение на начинът по който UTF-8 е разработен, стари конфигурационни файлове, шел-скриптове и дори много голяма част от по-стар софтуер могат да работят коректно с Unicode текст въпреки, че Unicode е разработен години след като те са били създадени.

Съществува **ASCII art** — форма на изобразително изкуство. При него се използват ASCII за представяне на изображения (фиг. 1). При създаването на такова изображение се използват символи на букви, цифри и пунктуационни знаци. В ASCII art се използват около 95 символа от ASCII таблицата. Това е така защото ASCII е национално представена таблица и останалите 160 символи отразяват конкретната кодова таблица. Това възпрепятства използването им в ASCII art пана. Първоначално ASCII art се е правело ръчно. Сега съществуват множество генератори на ASCII art. Тези програми автоматично създават ASCII изображения.

Таблица 1: ASCII символи

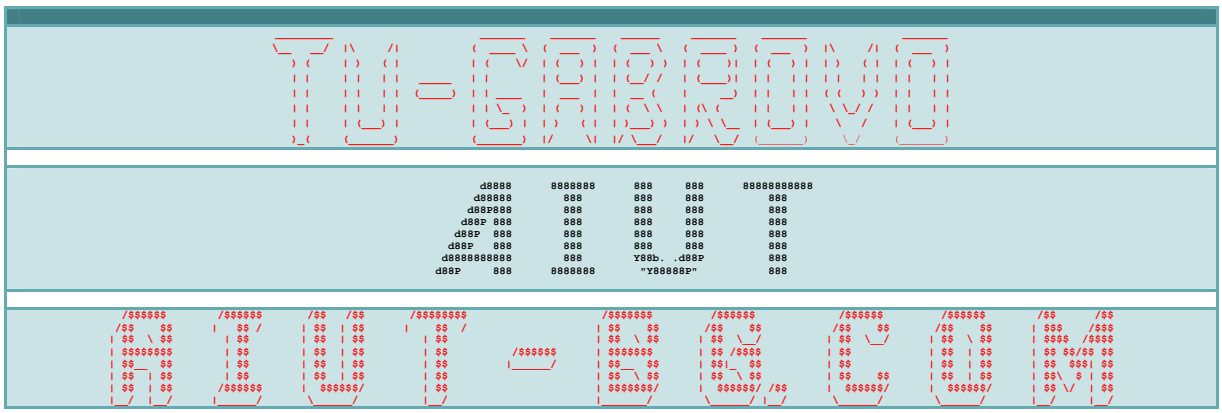
ASCII	Hex	Symbol	ASCII	Hex	Symbol	ASCII	Hex	Symbol	ASCII	Hex	Symbol
0	0	NUL	16	10	DLE	32	20	(space)	48	30	0
1	1	SOH	17	11	DC1	33	21	!	49	31	1
2	2	STX	18	12	DC2	34	22	"	50	32	2
3	3	ETX	19	13	DC3	35	23	#	51	33	3
4	4	EOT	20	14	DC4	36	24	$	52	34	4
5	5	ENQ	21	15	NAK	37	25	%	53	35	5
6	6	ACK	22	16	SYN	38	26	&	54	36	6
7	7	BEL	23	17	ETB	39	27	'	55	37	7
8	8	BS	24	18	CAN	40	28	(	56	38	8
9	9	TAB	25	19	EM	41	29	)	57	39	9
10	A	LF	26	1A	SUB	42	2A	*	58	3A	:
11	B	VT	27	1B	ESC	43	2B	+	59	3B	;
12	C	FF	28	1C	FS	44	2C	,	60	3C	<
13	D	CR	29	1D	GS	45	2D	-	61	3D	=
14	E	SO	30	1E	RS	46	2E	.	62	3E	>
15	F	SI	31	1F	US	47	2F	/	63	3F	?
ASCII	**Hex**	**Symbol**	**ASCII**	**Hex**	**Symbol**	**ASCII**	**Hex**	**Symbol**	**ASCII**	**Hex**	**Symbol**
64	40	@	80	50	P	96	60	`	112	70	p
65	41	A	81	51	Q	97	61	a	113	71	q
66	42	B	82	52	R	98	62	b	114	72	r
67	43	C	83	53	S	99	63	c	115	73	s
68	44	D	84	54	T	100	64	d	116	74	t
69	45	E	85	55	U	101	65	e	117	75	u
70	46	F	86	56	V	102	66	f	118	76	v
71	47	G	87	57	W	103	67	g	119	77	w
72	48	H	88	58	X	104	68	h	120	78	x
73	49	I	89	59	Y	105	69	i	121	79	y
74	4A	J	90	5A	Z	106	6A	j	122	7A	z
75	4B	K	91	5B	[	107	6B	k	123	7B	{
76	4C	L	92	5C	\	108	6C	l	124	7C	|
77	4D	M	93	5D	]	109	6D	m	125	7D	}
78	4E	N	94	5E	^	110	6E	n	126	7E	~
79	4F	O	95	5F	_	111	6F	o	127	7F	

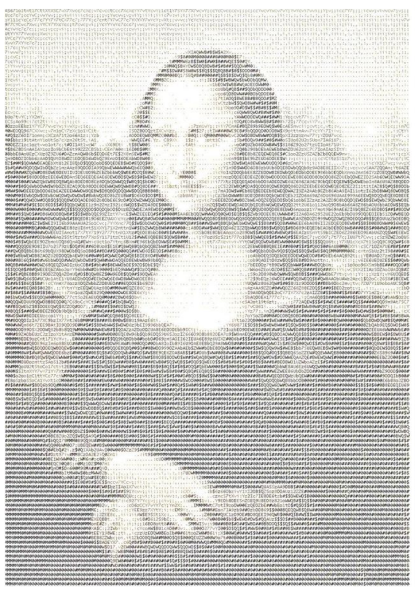
Таблица 2: ASCII управляващи символи

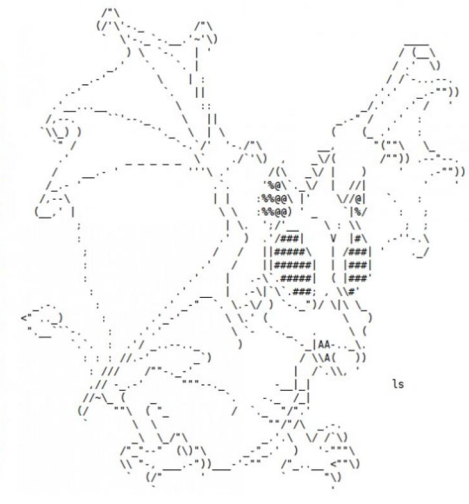
ASCII	Hex	Symbol	Type	Description
0	0	NUL		Null
1	1	SOH	CC	Start of Heading
2	2	STX	CC	Start of Text
3	3	ETX	CC	End of Text
4	4	EOT	CC	End of Transmission
5	5	ENQ	CC	Enquiry
6	6	ACK	CC	Acknowledge
7	7	BEL		Bell (audible or attention signal)
8	8	BS	FE	Backspace
9	9	TAB	FE	Horizontal Tabulation
10	A	LF	FE	Line Feed
11	B	VT	FE	Vertical Tabulation
12	C	FF	FE	Form Feed
13	D	CR	FE	Carriage Return
14	E	SO		Shift Out
15	F	SI		Shift In

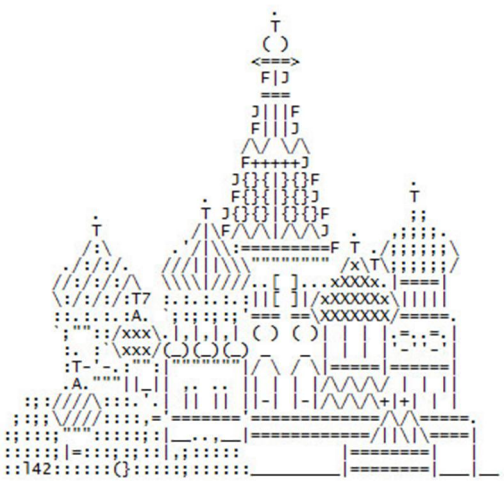
ASCII	Hex	Symbol	Type	Description
16	10	DLE	CC	Data Link Escape
17	11	DC1		Device Control 1
18	12	DC2		Device Control 2
19	13	DC3		Device Control 3
20	14	DC4		Device Control 4
21	15	NAK	CC	Negative Acknowledge
22	16	SYN	CC	Synchronous Idle
23	17	ETB	CC	End of Transmission Block
24	18	CAN		Cancel
25	19	EM		End of Medium
26	1A	SUB		Substitute
27	1B	ESC		Escape
28	1C	FS	IS	File Separator
29	1D	GS	IS	Group Separator
30	1E	RS	IS	Record Separator
31	1F	US	IS	Unit Separator

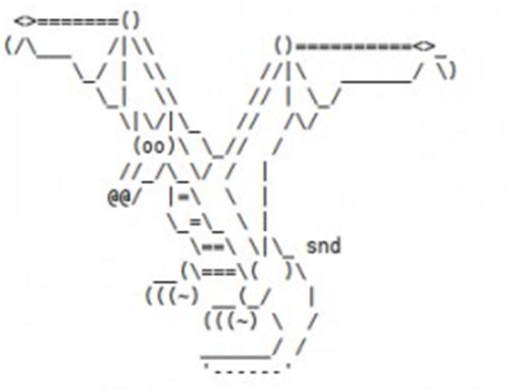
Type	Description
CC	Communication Control
FE	Format Effector
IS	Information Separator



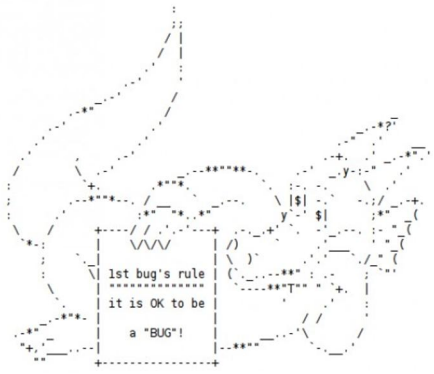


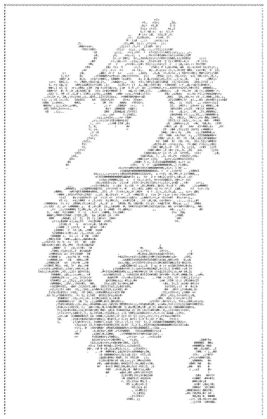


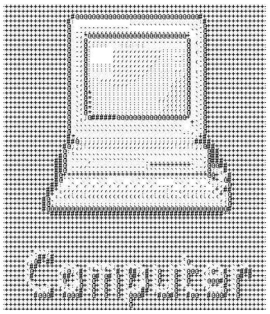


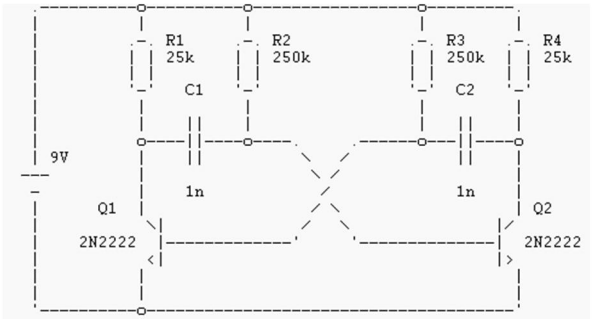

snd


Art by
Erik Andersson


lst bug's rule
it is OK to be
a "BUG"!






R1
25k
R2
250k
R3
250k
R4
25k
C1
C2
9V
1n
1n
Q1
Q2
2N2222
2N2222