Aula 16 - Cadeia de Caracteres

As famosas Strings =D

Contexto

- Se computadores só entendem números, como representar um texto em C ?
  - Se caracteres são símbolos e números são símbolos então caracteres podem ser representados por símbolos
- A linguagem C possui um tipo primitivo **char** utilizado para representar os caracteres através de números.
- Então, quais são os números utilizados para representar os caracteres?

Codificação

- Cada símbolo é associado a apenas um número
- Existem várias codificações como a ASCII, Unicode, etc.
- A codificação que utilizamos é a ASCII (American Standard Code for Information Interchange, 1960)
  - Símbolos:
    - Caracteres do alfabeto inglês
    - Dígitos
    - Símbolos de pontuação
    - Símbolos de espaçamento: espaço, tabulação, quebras de linha e de página
  - Representação:
    - Um byte (8 bits), onde o bit mais à esquerda é sempre 0
    - Inteiro entre 0 e 127

Dec	Hx	Oct	Char	Dec	Hx	Oct	Html	Chr	Dec	Hx	Oct	Html	Chr	Dec	Hx	Oct	Html	Chr
0	0	000	NUL (null)	32	20	040		Space	64	40	100	@	@	96	60	140	`	`
1	1	001	SOH (start of heading)	33	21	041	!	!	65	41	101	A	A	97	61	141	a	a
2	2	002	STX (start of text)	34	22	042	"	"	66	42	102	B	B	98	62	142	b	b
3	3	003	ETX (end of text)	35	23	043	#	#	67	43	103	C	C	99	63	143	c	c
4	4	004	EOT (end of transmission)	36	24	044	$	$	68	44	104	D	D	100	64	144	d	d
5	5	005	ENQ (enquiry)	37	25	045	%	%	69	45	105	E	E	101	65	145	e	e
6	6	006	ACK (acknowledge)	38	26	046	&	&	70	46	106	F	F	102	66	146	f	f
7	7	007	BEL (bell)	39	27	047	'	'	71	47	107	G	G	103	67	147	g	g
8	8	010	BS (backspace)	40	28	050	(	(	72	48	110	H	H	104	68	150	h	h
9	9	011	TAB (horizontal tab)	41	29	051	)	)	73	49	111	I	I	105	69	151	i	i
10	A	012	LF (NL line feed, new line)	42	2A	052	*	*	74	4A	112	J	J	106	6A	152	j	j
11	B	013	VT (vertical tab)	43	2B	053	+	+	75	4B	113	K	K	107	6B	153	k	k
12	C	014	FF (NP form feed, new page)	44	2C	054	,	,	76	4C	114	L	L	108	6C	154	l	l
13	D	015	CR (carriage return)	45	2D	055	-	-	77	4D	115	M	M	109	6D	155	m	m
14	E	016	SO (shift out)	46	2E	056	.	.	78	4E	116	N	N	110	6E	156	n	n
15	F	017	SI (shift in)	47	2F	057	/	/	79	4F	117	O	O	111	6F	157	o	o
16	10	020	DLE (data link escape)	48	30	060	0	0	80	50	120	P	P	112	70	160	p	p
17	11	021	DC1 (device control 1)	49	31	061	1	1	81	51	121	Q	Q	113	71	161	q	q
18	12	022	DC2 (device control 2)	50	32	062	2	2	82	52	122	R	R	114	72	162	r	r
19	13	023	DC3 (device control 3)	51	33	063	3	3	83	53	123	S	S	115	73	163	s	s
20	14	024	DC4 (device control 4)	52	34	064	4	4	84	54	124	T	T	116	74	164	t	t
21	15	025	NAK (negative acknowledge)	53	35	065	5	5	85	55	125	U	U	117	75	165	u	u
22	16	026	SYN (synchronous idle)	54	36	066	6	6	86	56	126	V	V	118	76	166	v	v
23	17	027	ETB (end of trans. block)	55	37	067	7	7	87	57	127	W	W	119	77	167	w	w
24	18	030	CAN (cancel)	56	38	070	8	8	88	58	130	X	X	120	78	170	x	x
25	19	031	EM (end of medium)	57	39	071	9	9	89	59	131	Y	Y	121	79	171	y	y
26	1A	032	SUB (substitute)	58	3A	072	:	:	90	5A	132	Z	Z	122	7A	172	z	z
27	1B	033	ESC (escape)	59	3B	073	;	;	91	5B	133	[	[	123	7B	173	{	{
28	1C	034	FS (file separator)	60	3C	074	<	<	92	5C	134	\	\	124	7C	174			|
29	1D	035	GS (group separator)	61	3D	075	=	=	93	5D	135	]	]	125	7D	175	}	}
30	1E	036	RS (record separator)	62	3E	076	>	>	94	5E	136	^	^	126	7E	176	~	~
31	1F	037	US (unit separator)	63	3F	077	?	?	95	5F	137	_	_	127	7F	177	&#127;	DEL

Source: www.LookupTables.com

Caracteres

- Existem três maneiras diferentes de inicializar uma variável do tipo char
  - Ex:
    - Inicializar uma variável char com a letra A maiúsculo (valor 65 na codificação na tabela ASCII):

```
char c = 65;      /* código ASCII */
char c = 'A';     /* caractere em aspas simples */
char c = '\065';  /* código ASCII em aspas simples, 0 a esquerda */
```

Caracteres

- Alguns caracteres especiais que geralmente começam com \:
  - Aspas simples : ‘\’
  - Contrabarra: ‘\\’
  - Quebra de linha: ‘\n’
  - Tabulação: ‘\t’

Expressões

- Como caracteres são números, podemos realizar operações com eles:

```
char c = 'A' + 1;   /* resulta em 'B' */
                    /* 65 (valor de 'A') + 1 resulta em 66 (valor de 'B') */
char c = 'A' < 'B'; /* resulta em verdadeiro (1) pois 65 < 66 */
```

- A leitura e escrita de caracteres é feita utilizando o identificador de tipo **%c**

```
#include <stdio.h>
int main( void )
{
    char i;
    scanf( "%c", i );
    for( i = 'A'; i <= 'Z'; ++i )
        printf( "%c ", i );
    return 0;
}
```

EXEMPLO

```c
#include <stdio.h>
#include <stdlib.h>
#include <string.h>


int main (){

    char palavra[11] = "Aula de ITP";
    int i;
    for(i = strlen(palavra)-1; i>=0; i--){
        printf("%c", palavra[i]);
    }
    printf("\n");
    return 0;
}
```

Representação do Texto

- Um texto em C é representado por um vetor de caracteres terminado pelo caracter ‘\0’
- O ‘\0’ chamado de **caracter nulo** é usado **somente** para delimitar o final de um texto. O caracter ‘\0’ é representado somente internamente, não sendo necessário adicioná-lo ao fim de uma cadeia de caracteres
- O texto ocupa um espaço de memória da seguinte maneira:
    - Se o texto tiver **N** símbolos, serão alocados **N+1 bytes**
    - O último símbolo ocupa a posição **N-1** do compartimento alocado e a última posição é ocupada pelo ‘\0’

Texto em C

- Como vimos, a linguagem C não tem um tipo primitivo que represente texto, mas as bibliotecas provêem funções para manipular as cadeias de caracteres.
- Existe uma sintaxe para representar textos em C
  - A sequência de símbolos deve ser delimitada por **“ ”**
  - Para imprimir um texto devemos usar o identificador **%s**

```c
#include <stdio.h>

int main( void )
{
    char texto[] = "Hello world!\n24/05/2013";
    printf( "%s\n", texto );
    puts( texto );
    return 0;
}
```

Bibliotecas Padrão

- stdio.h
  - int getchar (void) /*lê caractere*/
  - char putchar (char c) /*imprime um caractere*/
- ctype.h
  - char tolower (char c) /*transforma em minúsculo*/
  - char toupper (char c) /*transforma em maiúsculo*/
- string.h
  - char* strcat (char *str1, **const char** *str2) /*concatenação*/
  - int strcmp (**const char** *str1, **const char** *str2)/*comparação*/
  - char* strcpy (**char** *str1, **const char** *str2)/*cópia*/
  - size_t strlen (**const char** *str)/*calcula tamanho*/

Leitura de Textos

- scanf: lê enquanto não há espaço em branco

```
char v[20];
scanf( "%s", &v );
```

- gets: lê até encontrar uma nova linha ou o fim da entrada

```
char v[20];
gets( v );
```

- Ambas funções acrescentam o ‘\0’ ao final do vetor ao terminar a leitura