

MANIPULAÇÃO DE STRINGS

O TIPO DE DADO STRING

As strings, como trechos de texto, são os tipos de dados mais familiares aos seres humanos. O Pascal padrão não fornecia tipos de dados de String; tínhamos que utilizar apenas o tipo Char e, para trabalharmos com textos, devíamos utilizar um array de Char.

O Turbo Pascal, felizmente, oferece para os usuários o tipo de dado String. Embora este tipo seja considerado um array de Char, podemos ignorar este fato e utilizá-lo normalmente. Quando necessário, podemos utilizá-lo como um array (por exemplo, usando os colchetes com um índice para individualizar cada caracter do string). Observe o exemplo abaixo:

```
S := 'ASPER';  
write(s[4]); {será exibida a letra E, correspondente a 4ª letra da string S}
```

O tamanho de um string pode variar entre 0 e 255 caracteres. Se na declaração de uma variável string não especificarmos o tamanho máximo do string, ele assumirá 255. Por exemplo:

```
var  
S1 : string;  
S2 : string[10];
```

No exemplo acima, a variável S1 pode conter até 255 caracteres, enquanto a variável S2 pode conter no máximo 10 caracteres. Esse tamanho máximo nós denominamos de comprimento físico do string, que é o que determina o espaço reservado para a variável.

Se na variável S2 for armazenado um string de 4 caracteres, por exemplo, o comprimento físico continua sendo de 10 caracteres, enquanto que o espaço ocupado, no caso 4 caracteres, é o que denominamos comprimento lógico do string. O comprimento lógico de um string pode variar conforme o valor recebido pela variável durante o programa.

Resumindo, temos então que um string pode ter o seu comprimento físico variando de 1 a 255 caracteres, e o seu comprimento lógico variando de 0 até o valor do comprimento físico.

FUNÇÕES E PROCEDIMENTO PREDEFINIDOS

O Turbo Pascal dispõe de algumas funções e procedimentos que visam em essência, à otimização do trabalho do programador na parte que se refere à utilização de strings:

LENGTH	UPCASE	CONCAT
POS	COPY	DELETE
INSERT	VAL	STR
CHR	ORD	

LENGTH – Função que retorna o número de caracteres de uma string. Sua sintaxe é:

```
LENGTH (str : string) : byte;
```

Exemplo:

```
tam := length('TURBO PASCAL');  
writeln (tam); {será exibido o valor 12}
```

UPCASE – Função que retorna o caractere contido no parâmetro em maiúsculo. Sua sintaxe é:

```
UPCASE (ch : char) : char;
```

Exemplo:

```
letra := 'a';  
maiusc := upcase (letra);  
writeln (maiusc); {será exibida a letra 'A' (maiúscula) }
```

CONCAT – Função que retorna a união de duas ou mais strings passadas como parâmetros. Sua sintaxe é:

```
CONCAT (str1 , str2 , ... , strn : string) : string;
```

Exemplo:

```
pal1 := 'TURBO';  
pal2 := 'PASCAL';  
uniao := concat (pal1, ', ', pal2);  
writeln (uniao); {será exibido o string 'TURBO PASCAL'}
```

A função CONCAT tem efeito semelhante ao operador + (operador de concatenação).

Exemplo:

```
pal1 := 'TURBO';  
pal2 := 'PASCAL';  
uniao := pal1 + ' ' + pal2;  
writeln (uniao); {será exibido o string 'TURBO PASCAL'}
```

POS – Função que retorna a posição que uma substring ocupa dentro de uma string passadas como parâmetro. Sua sintaxe é:

```
POS (substr , str : string) : byte;
```

Exemplo:

```
frase := 'VAMOS ESTUDAR MAIS';  
pesq := 'ESTU';  
posicao := pos (pesq, frase);  
writeln (posicao); {será exibido o valor 7}
```

COPY – Função que retorna uma substring de uma string passadas como parâmetro, de acordo com sua posição e quantidade de caracteres especificados. Sua sintaxe é:

```
COPY (str:string; pos:byte; quant:byte) : string;
```

Exemplo:

```
frase := 'VAMOS ESTUDAR MAIS';  
pedaco := copy(frase,7,4);  
writeln (pedaco); {será exibido o string 'ESTU'}
```

DELETE – Procedimento que exclui um pedaço de uma string passada como parâmetro, de acordo com uma posição e quantidade de caracteres especificados. Sua sintaxe é:

```
DELETE (var str:string; pos:byte; quant:byte);
```

Exemplo:

```
frase := 'TURBO PASCAL 7.0';  
delete (frase,7,7);  
writeln (frase); {será exibido o string 'TURBO 7.0'}
```

INSERT – Procedimento que permite inserir uma substring dentro de uma string, em uma posição especificada. Sua sintaxe é:

```
INSERT (substr:string; var str:string; pos:byte);
```

Exemplo:

```
frase := 'CURSO DE INFORMATICA';  
insert ('MICRO',frase,10);  
writeln (frase); {será exibido o string 'CURSO DE MICROINFORMATICA'}
```

VAL – Procedimento que converte uma string passada como parâmetro para valor numérico. Caso o conteúdo da string não seja possível de ser convertido, o fato será informado em uma variável de retorno de erro. Se o retorno de erro for diferente de 0 (zero), implica que houve um erro de conversão, e este valor de retorno é a posição onde ocorreu o primeiro erro. Sua sintaxe é:

```
VAL (str:string; var num:integer|real; var erro:integer);
```

Exemplo 1:

```
codigo := '017348';  
val (codigo,numero,erro);  
writeln (numero); {será exibido o valor 17348}  
writeln (erro); {será exibido o valor 0}
```

Exemplo 2:

```
codigo := '12X345'  
val (codigo,numero,erro);  
writeln (erro) {será exibido o valor 3}
```

STR – Procedimento que converte uma variável numérica em um string, determinando o tamanho do string e a quantidade de casas decimais. Sua sintaxe é:

```
STR (num [:tam [:dec]]; var str:string);
```

Exemplo:

```
numero := 12.3;
```

```
str (numero:6:2,conv);  
writeln (conv); {será exibido o string ' 12.30'}
```

CHR – Função que retorna o caracter correspondente ao valor ASCII especificado. Sua sintaxe é:

```
CHR (codigo:byte) : char;
```

Exemplo:

```
    codigo := 65;  
caracter := CHR(codigo);  
writeln (caracter); {será exibido o caracter 'A'}
```

ORD – Função que retorna o valor ASCII correspondente ao caracter especificado. Sua sintaxe é:

```
ORD (caracter:char) : byte;
```

Exemplo:

```
    caracter := 'A';  
codigo := ORD(caracter);  
writeln (codigo); {será exibido 65}
```

ANEXO – TABELA ASC II

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		DEL

Source: www.asciitable.com

128	Ç	144	É	160	á	176	⋮	193	⊥	209	ƒ	225	ß	241	±
129	ù	145	æ	161	í	177	⋮	194	⊥	210	π	226	Γ	242	≥
130	é	146	Æ	162	ó	178	⋮	195	⊥	211	ℓ	227	π	243	≤
131	â	147	ô	163	ú	179		196	—	212	ℓ	228	Σ	244	∫
132	ä	148	ö	164	ÿ	180	†	197	†	213	ƒ	229	σ	245	∫
133	à	149	ò	165	ÿ	181	†	198	†	214	π	230	μ	246	÷
134	â	150	û	166	•	182	‡	199	‡	215	‡	231	τ	247	≈
135	ç	151	ù	167	°	183	π	200	ℓ	216	≠	232	Φ	248	°
136	ê	152	—	168	¿	184	¶	201	¶	217	∫	233	⊙	249	·
137	ë	153	Ö	169	—	185	‡	202	≠	218	∫	234	Ω	250	·
138	è	154	Û	170	¬	186	‡	203	¶	219	■	235	δ	251	√
139	ï	156	£	171	½	187	¶	204	‡	220	■	236	∞	252	—
140	î	157	¥	172	¼	188	‡	205	=	221	■	237	φ	253	²
141	ï	158	—	173	¡	189	‡	206	‡	222	■	238	e	254	■
142	Ä	159	f	174	«	190	∫	207	⊥	223	■	239	∧	255	
143	Å	192	L	175	»	191	∫	208	≠	224	α	240	≡		

Source: www.asciitable.com