ORGANIZAÇÃO DE COMPUTADORES DIGITAIS

Exercício Programa

Fernando Karchiloff Gouveia de Amorim – nUSP 10387644
Lucas Pereira Castelo Branco – nUSP 10258772

PROBLEMA PROPOSTO

a) Escreva uma função com o protótipo void converte (char ch, int *tipo, char *valor); que recebe um caractere ch e devolve em *tipo 0, se o caractere for um número inteiro, 1 se for uma letra (maiúscula ou minúscula) e 2 caso contrário; e além disso, no caso de ser uma letra, converte para maiúscula, senão devolve ch inalterado.

b) Escreva um programa que leia uma sequência de n caracteres e imprima a sequência convertida para maiúscula, eliminando os caracteres que não forem letras ou números.

COMENTÁRIOS

- A assinatura do método exigido em a) referenciava a ponteiros
  - Escolha de C como linguagem de alto nível a ser implementada.
- O ponto chave do problema é COMO seria feito o processo da identificação do tipo de um caractere
  - A letra b) é dependente direta da letra a).

CÓDIGO EM ALTO NÍVEL – LETRA A

- Para o código em C, temos duas opções de implementação para a letra a), que serão demonstradas a seguir!
- Na primeira delas, optamos por usar da biblioteca <ctype.h>
  - Possui métodos que fazem a verificação direta de um dado caractere (ou inteiro) e um tipo, e retorna um booleano.
  - Possui métodos que modificam o dado caractere, como a conversão de minúsculo para maiúsculo solicitada.

OPÇÃO 1 – CTYPE.H

```c
void converte(char ch, int *tipo, char *valor){
    //converte de char para inteiro
    int c = (int) ch;
    //garante que valor receba o valor de ch
    *valor = ch;
    //verifica se é alfanumérico
    if(isalnum(c)){
        if(isdigit(c)){
            //é um número inteiro, logo retorna o tipo 0
            *tipo = 0;
            return;
        }else{
            //é uma letra, logo, retorna o tipo 1
            *tipo = 1;
            if(islower(c)){
                *valor = toupper(c);
            }
            return;
        }
    }else{
        //não é um número nem uma letra, logo, retorna o tipo 2
        *tipo = 2;
        return;
    }
}
```

OPÇÃO 2- ASCII

- A outra forma encontrada fora verificando o valor inteiro de um caractere e comparando-a com a **Tabela ASCII**

- **ASCII,** ou American Standard Code for Information Interchange, é o padrão computacional de caracteres latinos.

- Portanto, a partir dos intervalos da tabela, conseguimos definir o tipo.

TABELA ASCII

ASCII control characters

00	NULL	(Null character)
01	SOH	(Start of Header)
02	STX	(Start of Text)
03	ETX	(End of Text)
04	EOT	(End of Trans.)
05	ENQ	(Enquiry)
06	ACK	(Acknowledgement)
07	BEL	(Bell)
08	BS	(Backspace)
09	HT	(Horizontal Tab)
10	LF	(Line feed)
11	VT	(Vertical Tab)
12	FF	(Form feed)
13	CR	(Carriage return)
14	SO	(Shift Out)
15	SI	(Shift In)
16	DLE	(Data link escape)
17	DC1	(Device control 1)
18	DC2	(Device control 2)
19	DC3	(Device control 3)
20	DC4	(Device control 4)
21	NAK	(Negative acknowl.)
22	SYN	(Synchronous idle)
23	ETB	(End of trans. block)
24	CAN	(Cancel)
25	EM	(End of medium)
26	SUB	(Substitute)
27	ESC	(Escape)
28	FS	(File separator)
29	GS	(Group separator)
30	RS	(Record separator)
31	US	(Unit separator)
127	DEL	(Delete)

ASCII printable characters

32	space	64	@	96	`
33	!	65	A	97	a
34	"	66	B	98	b
35	#	67	C	99	c
36	$	68	D	100	d
37	%	69	E	101	e
38	&	70	F	102	f
39	'	71	G	103	g
40	(	72	H	104	h
41	)	73	I	105	i
42	*	74	J	106	j
43	+	75	K	107	k
44	,	76	L	108	l
45	-	77	M	109	m
46	.	78	N	110	n
47	/	79	O	111	o
48	0	80	P	112	p
49	1	81	Q	113	q
50	2	82	R	114	r
51	3	83	S	115	s
52	4	84	T	116	t
53	5	85	U	117	u
54	6	86	V	118	v
55	7	87	W	119	w
56	8	88	X	120	x
57	9	89	Y	121	y
58	:	90	Z	122	z
59	;	91	[	123	{
60	<	92	\	124	|
61	=	93	]	125	}
62	>	94	^	126	~
63	?	95	_		

Extended ASCII characters

128	Ç	160	á	192	└	224	Ó
129	ü	161	í	193	┴	225	ß
130	é	162	ó	194	┬	226	Ô
131	â	163	ú	195	├	227	Ò
132	ä	164	ñ	196	─	228	õ
133	à	165	Ñ	197	┼	229	Õ
134	å	166	ª	198	ã	230	µ
135	ç	167	º	199	Ã	231	þ
136	ê	168	¿	200	╚	232	Þ
137	ë	169	®	201	╔	233	Ú
138	è	170	¬	202	╩	234	Û
139	ï	171	½	203	╦	235	Ù
140	î	172	¼	204	╠	236	ý
141	ì	173	¡	205	═	237	Ý
142	Ä	174	«	206	╬	238	¯
143	Å	175	»	207	¤	239	´
144	É	176	░	208	ð	240	≡
145	æ	177	▒	209	Ð	241	±
146	Æ	178	▓	210	Ê	242	‗
147	ô	179	│	211	Ë	243	¾
148	ö	180	┤	212	È	244	¶
149	ò	181	Á	213	ı	245	§
150	û	182	Â	214	Í	246	÷
151	ù	183	À	215	Î	247	¸
152	ÿ	184	©	216	Ï	248	°
153	Ö	185	╣	217	┘	249	¨
154	Ü	186	║	218	┌	250	·
155	ø	187	╗	219	█	251	¹
156	£	188	╝	220	▄	252	³
157	Ø	189	¢	221	¦	253	²
158	×	190	¥	222	Ì	254	■
159	ƒ	191	┐	223	▀	255	nbsp

OPÇÃO 2- ASCII

```c
void converte(char ch, int *tipo, char *valor){
    /* O tipo 'char' eh na verdade uma representacao de um
       inteiro, transformado para caracter, exemplos:
       57 -> '9'
       48 -> '0'
       97 -> 'a'*/
    int caracter = (int) ch;

    //if(ch) inteiro retorna tipo 0
    if(caracter >= 48 && caracter <= 57){ //Inteiro
        *tipo = 0;
        *valor = (char) caracter;
    }
    else{
        //if(ch) letra(M ou m) retorna tipo 1
        /*
            No caso de ser letra, converte para maiuscula,
            se nao devolve inalterado. */
        if((caracter >= 65 && caracter <= 90) || ( //Letra
            caracter >= 97 && caracter <= 122)){

            *tipo = 1;
            if(caracter >= 97 && caracter <= 122){ //Se for minuscula
                /*
                    32 eh o valor correspondente na Tabela ASCII
                    para transformar de minuscula para maiuscula
                */
                caracter = caracter - 32;
            }
            *valor = (char)caracter;
        }
        else{   //Outra coisa.
            //else retorna tipo 2
            *tipo = 2;
            *valor = ch;
        }

    }
}
```

CÓDIGO EM ALTO NÍVEL – LETRA B

- A partir de qualquer uma das implementações da letra a), a letra b) seria válida, pois a dependência entre eles é apenas de obter o tipo pelo ponteiro e o possível valor alterado.

- Assim, a implementação será:

LETRA B

```c
void imprimeSequencia(char array[]){
    /*
        Le uma sequencia de n caracteres e imprime
        a sequencia convertida para maiscula, eliminando
        caracteres que nao forem letras ou numeros.
    */
    printf("Imprime sequencia.\n");
    int n = strlen(array);  //uso de string requer <string.h>
    //Caso seja um array de caracteres separados, pode se usar sizeof().
    int tipo;
    char valor;
    for(int i=0 ;i < n; i++){
        converte(array[i],&tipo,&valor);  //Converte
        if(tipo == 1 || tipo == 0){  //Se for letra, imprime (ja convertido)
            printf("%c",valor);
        }
    }
}
```

CÓDIGO EM ASSEMBLY

- Para o código em Assembly, bastava fazermos um processo semelhante ao encontrado na segunda opção da letra A
  - Ou seja, precisávamos somente do valor inteiro correspondente do inteiro, e compará-lo com os intervalos
  - A partir dos testes lógicos, poderíamos fazer tanto a conversão (item a)) quanto a impressão (item b).

- Dessa forma, unificando as rotinas envolvidas:

INICIO DO CÓDIGO- ASSEMBLY

```
#Problema 10 - Conversao de caracteres e impressao de uma sequencia de caracteres sendo convertido
#Dupla: Lucas Pereira e Fernando Karchiloff
#
#Tipos de registradores utilizados:
#       $a0   - recebe caracter a ser convertido
#       $a1   - contador que incrementa ao percorrer a string
#       $a2   - armazena o tamanho da string
#       $t0   - usado para os testes logicos
#       $t1   - usada para os testes logicos
#       $t2   - recebe o tipo do caracter  (0, 1, 2)
#       $t3   - recebe o byte do caracter
```

```
.text
.globl main
main:
    li $a1,0       #carrega contador apartir do 0
    lb $a2,$tam  #carrega o tamanho da string
    j imprime_sequencia   #vai para a funcao de imprimir sequencia

imprime_sequencia:   #funcao que imprime uma sequencia de caracteres (convertendo)
    lb $t3,$sequencia($a1)       #pega o byte segundo o indice apresentado
    move $a0, $t3                #passa o caracter para o $a0 como argumento
    addi $a1,$a1,1                #adiciona 1 no contador
    jal converte                     #funcao de conversao
    jal printa_caracter       #funcao de printar caracter
    blt $a1,$a2, imprime_sequencia   #se a1 menor que a2 faz loop
    j fim_programa              #chama o encerramento do programa

fim_programa:   #funcao que imprime uma sequencia de caracteres
    li $v0, 10    #codigo para sair do programa
    syscall   #chama o sistema operacional
```

```
converte:  #funcao que converte um unico caracter

    #numero
    sgeu $t0, $a0, 48   #intervalo de ascii
    sleu $t1, $a0, 57   #intervalo de ascii
    and $t0, $t0, $t1   #se esta dentro desse intervalo  $t0 = 1 / c.c. $t0 = 0
    beq $t0, 1, numero  #se eh numero entra na funcao

    #letra maiuscula
    sgeu $t0, $a0, 65   #intervalo de ascii
    sleu $t1, $a0, 90   #intervalo de ascii
    and $t0, $t0, $t1   #se esta dentro desse intervalo $t0 = 1 / c.c. $t0 = 0
    beq $t0, 1, letra_maiuscula  #se eh letra maiuscula entra na funcao

    #letra minuscula
    sgeu $t0, $a0, 97     #intervalo de ascii
    sleu $t1, $a0, 122    #intervalo de ascii
    and $t0, $t0, $t1     #se entra dentro desse intervalo $t0 = 1 / c.c. $t0 = 0
    beq $t0, 1, letra_minuscula  #se eh letra minuscula entra na funcao

    #se passou direto entra para caracter especial
    j carac_especial  #funcao de caracter especial

converte_retorno:   #funcao que entra para dar jump ao endereco de retorno
    jr $ra     #da jump para a instrucao onde o registro de retorno aponta
```

```
printa_caracter:   #funcao que faz o print dos caracteres
    beq $t2, 2, printa_carac_retorno #se nao for letra ou numero, ja retorna sem printar
    li $v0, 11   #printa caracter
    syscall    #chama o sistema para fazer a operacao
    j printa_carac_retorno   #chama o retorno ao final para voltar ao loop

printa_carac_retorno:  #funcao que entra para dar jump ao endereco de retorno
    jr $ra     #da jump para a instrucao onde o registro de retorno aponta

numero:      #funcao que coloca o tipo do caracter no registrador $t2
    li $t2, 0    #coloca 0 em $t2 para numeros
    j converte_retorno     #chama o retorno da funcao de conversao

letra_minuscula:    #funcao que coloca o tipo do caracter no registrador $t2 e converte em maiuscula
    li $t2, 1     #coloca 1 em $t2 para letras
    sub $a0, $a0, 32   #transforma para maiuscula subtraindo 32
    j converte_retorno   #chama o retorno da funcao de conversao

letra_maiuscula:   #funcao que coloca o tipo do caracter no registrador $t2
    li $t2, 1    #coloca 1 em $t2 para letras
    j converte_retorno   #chama o retorno da funcao de conversao

carac_especial:    #funcao que coloca o tipo do caracter no registrador $t2
    li $t2, 2    #coloca 2 em $t2 para caracteres especiais
    j converte_retorno    #chama o retorno da funcao de conversao

.data
#deve ser especificado o tamanho da sequencia de caracteres em .asciiz e a sequencia em si
$tam: .word 14        #tamanho da sequencia de caracteres
$sequencia: .asciiz "C0oNv3eRS 4Ao"
```