Algoritmos e Programação I
Exercício-Programa 3

Marcelo Hashimoto

19 de março de 2013

1 Introdução

Neste Exercício-Programa você vai implementar um *codificador e decodificador hexadecimal*, baseado principalmente nos conceitos de *vetores* e *strings*. Não é permitido o uso de qualquer conceito que tenha sido ensinado depois da Semana 11 de aula.

Parte da correção deste exercício será feita *automaticamente*, portanto é fundamental que a entrada e a saída do seu programa sejam *exatamente* como a especificação pede.

2 Caracteres como inteiros

Conceitualmente, variáveis do tipo `char` armazenam caracteres como `'a'`, `'b'` e `'c'`. Tecnicamente, no entanto, o que elas armazenam são *códigos numéricos* que representam caracteres. Você pode facilmente descobrir que códigos são esses se atribuir o valor de um `char` a um `int`.

```
char c;
int i;

c = 'a';
i = c;

printf("%d\n", i);
```

Se você executar o programa acima, poderá verificar que ele imprime o código `97`. Substituindo `'a'` por `'b'`, ele imprime `98`. Os códigos para todas os caracteres do alfabeto latino podem ser vistos na tabela abaixo.

A	65	N	78	a	97	n	110
B	66	O	79	b	98	o	111
C	67	P	80	c	99	p	112
D	68	Q	81	d	100	q	113
E	69	R	82	e	101	r	114
F	70	S	83	f	102	s	115
G	71	T	84	g	103	t	116
H	72	U	85	h	104	u	117
I	73	V	86	i	105	v	118
J	74	W	87	j	106	w	119
K	75	X	88	k	107	x	120
L	76	Y	89	l	108	y	121
M	77	Z	90	m	109	z	122

3 Inteiros como caracteres

Repare que todos os códigos numéricos acima estão entre 0 e 255. Para qualquer inteiro n nesse intervalo, existem dois inteiros, x e y, entre 0 e 15 tais que

$$n = 16x + y.$$

Descobrir esses dois inteiros é fácil: ao dividir n por 16, obtemos x como quociente e y como resto. Considere agora a seguinte representação dos inteiros entre 0 e 15 como caracteres:

0	representado por	`'0'`;	8	representado por	`'8'`;
1	representado por	`'1'`;	9	representado por	`'9'`;
2	representado por	`'2'`;	10	representado por	`'A'`;
3	representado por	`'3'`;	11	representado por	`'B'`;
4	representado por	`'4'`;	12	representado por	`'C'`;
5	representado por	`'5'`;	13	representado por	`'D'`;
6	representado por	`'6'`;	14	representado por	`'E'`;
7	representado por	`'7'`;	15	representado por	`'F'`.

Podemos concluir que, sob o esquema acima, qualquer inteiro entre 0 e 255 pode ser representado por dois caracteres dentro do conjunto $\mathcal{H} = \{$`'0'`, `'1'`, `'2'`, `'3'`, `'4'`, `'5'`, `'6'`, `'7'`, `'8'`, `'9'`, `'A'`, `'B'`, `'C'`, `'D'`, `'E'`, `'F'`$\}$.

4 Codificação

A partir das duas seções anteriores, considere a seguinte sequência de transformações:

caractere latino $\rightarrow$ inteiro entre 0 e 255 $\rightarrow$ dois inteiros entre 0 e 15 $\rightarrow$ dois caracteres em $\mathcal{H}$.

Alguns exemplos dessa sequência são dados a seguir:

`'A'`	$\rightarrow$	65	$\rightarrow$	4	e	1	$\rightarrow$	`'4'` e `'1'`
`'R'`	$\rightarrow$	82	$\rightarrow$	5	e	2	$\rightarrow$	`'5'` e `'2'`
`'i'`	$\rightarrow$	105	$\rightarrow$	6	e	9	$\rightarrow$	`'6'` e `'9'`
`'z'`	$\rightarrow$	122	$\rightarrow$	7	e	10	$\rightarrow$	`'7'` e `'A'`

5 Decodificação

Analogamente, podemos considerar a sequência inversa de transformações:

dois caracteres em $\mathcal{H}$ $\rightarrow$ dois inteiros entre 0 e 15 $\rightarrow$ inteiro entre 0 e 255 $\rightarrow$ caractere latino.

Exemplos análogos aos anteriores são dados a seguir:

`'4'` e `'1'`	$\rightarrow$	4	e	1	$\rightarrow$	65	$\rightarrow$	`'A'`
`'5'` e `'2'`	$\rightarrow$	5	e	2	$\rightarrow$	82	$\rightarrow$	`'R'`
`'6'` e `'9'`	$\rightarrow$	6	e	9	$\rightarrow$	105	$\rightarrow$	`'i'`
`'7'` e `'A'`	$\rightarrow$	7	e	10	$\rightarrow$	122	$\rightarrow$	`'z'`

6 Especificação

Seu programa deve executar **exatamente** as seguintes operações.

1. Imprime a string `"Digite 1 para codificar e 2 para decodificar: "`. Deve haver exatamente um espaço no final e não deve haver quebra de linha.

2. Recebe um inteiro do usuário através da função `scanf` com `%d`. Você pode supor que o usuário de fato digitará 1 ou 2, sem se preocupar em repetir o pedido ou corrigir a entrada.

3. Executa o código a seguir, que será explicado em aula:

```
do {
  scanf("%c", &c);
} while(c != '\n');
```

4. Imprime a string `"Digite o texto: "`. Deve haver exatamente um espaço no final e não deve haver quebra de linha.

5. Recebe uma linha de texto do usuário através da função `fgets`.

   (a) Se o inteiro recebido no Item 2 foi 1, você pode supor que o usuário digitará uma linha de texto com no máximo 25 caracteres *sem contar a quebra de linha*. Além disso, pode supor que cada um desses caracteres será ou um dos 52 latinos da Seção 2 ou um dos seis abaixo.

	32	,	44
!	33	.	46
"	34	?	63

   (b) Se o inteiro recebido no Item 2 foi 2, você pode supor que o usuário digitará uma linha de texto com no máximo 50 caracteres *sem contar a quebra de linha*. Além disso, pode supor que cada um desses caracteres será do conjunto $\mathcal{H}$ e que o total será par.

6. Imprime uma linha de texto.

   (a) Se o inteiro recebido no Item 2 foi 1, essa linha impressa é a *codificação* da linha que o usuário digitou, segundo o processo descrito na Seção 4. A quebra de linha deve ser desconsiderada.

   (b) Se o inteiro recebido no Item 2 foi 2, essa linha impressa é a *decodificação* da linha que o usuário digitou, segundo o processo descrito na Seção 5. A quebra de linha deve ser desconsiderada.

7. Termina.

7 Exemplos

Seguem dois exemplos de execução. Os trechos em vermelho indicam entrada do usuário.

```
Digite 1 para codificar e 2 para decodificar: 1
Digite o texto: Hello World!
48656C6C6F20576F726C6421

Digite 1 para codificar e 2 para decodificar: 2
Digite o texto: 48656C6C6F20576F726C6421
Hello World!
```

8 Detalhes de entrega

- Este exercício deve ser feito individualmente ou em grupo de 2 a 3 alunos.
- O programa deve ser entregue pelo Blackboard até as **23:50** do dia **11 de maio**.
- Entregue apenas um arquivo de código-fonte, como anexo e com extensão `c`.

O enunciado foi projetado para um prazo menor do que o estabelecido acima. Você pode entregar em cima da hora, mas **problemas técnicos do Blackboard não serão aceitos como justificativa para atrasos.**

9 Critérios de correção

Ao entregar seu trabalho, verifique se nenhum dos problemas abaixo ocorre.

- **Grupo com mais de 3 alunos:** nota zero.
- **Entrega atrasada:** nota zero.
- **Arquivos a mais:** nota zero.
- **Entrega não é anexo:** nota zero.
- **Entrega não tem extensão `c`:** nota zero.
- **Erro durante compilação:** nota zero.

- **Aviso durante compilação:** desconto de `1.0` ponto por aviso.
- **Uso de conceito ensinado depois da Semana** 11**:** nota zero.
- **Uso de conceito que não foi ensinado:** nota zero.
- **Não cumpre requisitos do enunciado:** nota zero.

Se qualquer tipo de plágio for constatado, **todos** os membros de **todos** os grupos envolvidos **no mínimo** receberão nota zero. Outras punições adicionais ainda poderão ser estabelecidas posteriormente pela coordenação.