PUC/Pr — Pontifícia Universidade Católica do Paraná
Curso de Bacharelado em Informática — 11/02/2019 - 12:35:14.4
Estruturas de Dados e Arquivos — Prof. Dr. P. Kantek
Exercícios de codificação — VIVO450a V: 3.67
1
Exercício : 1- ______ / ______ / ______

Bits e Bytes

Os computadores usam lógica binária. Vai daí que suas memórias organizam-se em dígitos binários (bits=BInary digiT). Assim, quando olha-se um conteúdo qualquer, o que se vê são apenas zeros e uns. É o chamado conteúdo binário. Ele não é muito prático, pois sempre é extenso (o número $1.300.000_{(10)}$ é igual a $100111101011000100000_{(2)}$ em binário) razão pela qual surgiu a representação hexadecimal, que é uma abreviatura do código binário. Agora cada 4 bits são representados por um único dígito hexadecimal. Com isso, o comprimento do conteúdo é 4 vezes menor. O mesmo $1.300.000_{(10)}$ agora é $13D620_{(16)}$. Juntando 8 bits (2 dígitos hexadecimais) tem-se um BYTE. Portanto o número $1.300.000_{(10)}$ ocupa 3 bytes, a saber: X'13.D6.20'.

Codificação e decodificação

Humanos não têm muita habilidade para lidar com conteúdos binários, nem mesmo hexadecimais. Assim, surgiram os códigos que permitem traduzir conteúdos humanos (letras e números decimais) em conteúdos computadorizados (binários). Há muitos códigos distintos. Este exercício vai lidar com o ASCII e com dois códigos inventados: Ortelino e Trocaletra.

O código ASCII utiliza 8 bits para representar cada caracter. Este código (até pelo seu nome) é um código padrão e é reconhecido por todos os equipamentos e softwares que o manuseiam. Entretanto, nada impede, que:

- outros códigos de 8 bits/caractere existam e sejam usados (por exemplo EBCDIC ou Z-CODE).
- outros códigos com número diferente de 8 bits/caractere sejam usados. Assim, tem-se o UNICODE e o ORTELINO (com 5 bits/caractere). Note-se que quanto maior o número de bits, mais caracteres podem ser representados. O UNICODE tem espaço para mais de um milhão de caracteres, seguramente mais do é necessário para representar todos os caracteres em todos os alfabetos em uso aqui na Terra. Da mesma maneira, ORTELINO só consegue representar 32 caracteres distintos.
- Finalmente, nada impede que se tenham códigos que representam os caracteres em um número *variável* de bits. O que se paga em aumento de complexidade no manuseio, se ganha em economia, já que nestes casos, os caracteres mais freqüentes usarão menos bits para serem representados. Uma das encarnações do UNICODE (UTF-8) é assim. O código TROCAELETRA é outro exemplo de código deste tipo.

Suponha o código de caracteres (ASCII convencional):

```
A=X'41'  H=X'48'  O=X'4F'  V=X'56'  c=X'63'  j=X'6A'  q=X'71'  x=X'78'
B=X'42'  I=X'49'  P=X'50'  W=X'57'  d=X'64'  k=X'6B'  r=X'72'  y=X'79'
C=X'43'  J=X'4A'  Q=X'51'  X=X'58'  e=X'65'  l=X'6C'  s=X'73'  z=X'7A'
D=X'44'  K=X'4B'  R=X'52'  Y=X'59'  f=X'66'  m=X'6D'  t=X'74'   =X'20'
E=X'45'  L=X'4C'  S=X'53'  Z=X'5A'  g=X'67'  n=X'6E'  u=X'75'
F=X'46'  M=X'4D'  T=X'54'  a=X'61'  h=X'68'  o=X'6F'  v=X'76'
G=X'47'  N=X'4E'  U=X'55'  b=X'62'  i=X'69'  p=X'70'  w=X'77'
```

A especificação X'zz', indica que os dígitos zz representam um número hexadecimal, correspondente a 8 bits (sendo 4 + 4).

Suponha a tabela binária:

```
0=0000, 1=0001, 2=0010, 3=0011, 4=0100, 5=0101, 6=0110, 7=0111
8=1000, 9=1001, A=1010, B=1011, C=1100, D=1101, E=1110, F=1111
```

Faça as seguintes conversões:

1. De Hexadecimal para caracteres

```
Exemplo:
X'4F 75 76 69 72 61 6D 20 64 6F 20 49 70 69 72 61 6E 67 61'
Ouviram do Ipiranga

4E6120726564652065207065697865207665 6A61
```

:____.

2. De Binário para caracter

```
Exemplo:
B'0100 0010 0110 0001 0110 0011 0110 0001 0110 1110 0110 0001'
Bacana

0100001001101111011011100110010101100011 01100001
```

:____.

Suponha agora o código ORTELINO

```
A=B'00000' F=B'00101' K=B'01010' P=B'01111' U=B'10100' Z=B'11001'
B=B'00001' G=B'00110' L=B'01011' Q=B'10000' V=B'10101'  =B'11010'
C=B'00010' H=B'00111' M=B'01100' R=B'10001' W=B'10110' -=B'11011'
D=B'00011' I=B'01000' N=B'01101' S=B'10010' X=B'10111' ,=B'11100'
E=B'00100' J=B'01001' O=B'01110' T=B'10011' Y=B'11000' .=B'11101'
```

3. Converta de caracteres para ORTELINO e depois represente em hexadecimal

```
Exemplo:
IVO VIU A UVA : X'45 5D AA A2 9A 06 A9 50 00', com 65 bits

AZAR DELES / tam= 50
```

Represente os últimos 3 bytes apenas: :________, ________,________.

4. Converta de ORTELINO (em hexa) para caractere

```
Exemplo:
X'72 34 D7 22 5A 04 28 80 com 60 bits : OI NOIS AQUI

tam= 50 / 02EA8D54911900
```

:____.

Suponha agora o código TROCALETRA (de exemplo, igual para todos)

```
B=B'0001     '    L=B'1001     '    X=B'00001    '    M=B'000001  '
A=B'0010     '    O=B'1010     '    H=B'01000    '    G=B'100000  '
P=B'0011     '    E=B'1011     '     =B'01001    '    R=B'100001  '
U=B'0101     '    D=B'1101     '    Q=B'10001    '    N=B'0000001 '
Z=B'0110     '    S=B'1110     '    V=B'11000    '    F=B'00000000'
I=B'0111     '    T=B'1111     '    J=B'11001    '    C=B'00000001'
```

O que significa: X'01.28.48.0C.15.C0' com 43 bits ? :______________.

Mais um, para você ver se entendeu: X'DA 92 86 F8', com 30 bits ? R='DOLARES'.

Suponha agora **OUTRO TROCALETRA**, que é o que você deve usar no seu exercício, e que É DIFERENTE PARA CADA FOLHA. (sugestão: risque o Trocaletra do exemplo acima, para não se confundir).

```
B=B'0001     '    N=B'1001     '    A=B'00001    '    V=B'000001  '
S=B'0010     '    P=B'1010     '    J=B'01000    '    D=B'100000  '
Q=B'0011     '    I=B'1011     '    G=B'01001    '    O=B'100001  '
 =B'0101     '    L=B'1101     '    Z=B'10001    '    U=B'0000001 '
M=B'0110     '    E=B'1110     '    F=B'11000    '    C=B'00000000'
H=B'0111     '    T=B'1111     '    X=B'11001    '    R=B'00000001'
```

5. Converta de TROCALETRA (em hexa) para caractere

```
Exemplo:
0002037F6210 / tam= 44 - CURITIBA

tam= 49 / 4F6860EB010900
```

:____.

6. Converta de caractere para TROCALETRA e depois represente em hexadecimal

```
Exemplo:
PARANA / tam= 31 -  A0808642

XICARA BOA / tam= 54
```

Represente os últimos 3 bytes apenas: ________, ________,________.

Observação

Perceba que QUALQUER computador desses que há por aí, escreve 8 bits quando quer gravar (ou ler) alguma coisa. Quando 1 caracter = 8 bits não ocorre nenhuma dificuldade. Por isso esse número (8) foi o escolhido.

Assim, quando se codifica em ASCII (ou em EBCDIC ou em qualquer código fixo com comprimento de 8 bits) **não** há necessidade de especificar a quantidade de bits utilizados na escrita. Quando a escrita acabar, acabou o código. Isto não é verdade quando se utiliza um código que tenha algo diferente de 8 bits por caracter. Por exemplo, se se utilizam 5 bits por caracter, ao escrever um único caracter usam-se 5 bits, mas não há como escrever apenas 5 bits em qualquer computador. Vai daí que se complementa o código com 3 zeros à direita (*stuffing bits*) e escrevem-se como sempre 8 bits por caracter. Mas, há necessidade de avisar que dos 8 bits escritos apenas 5 interessam.

Com muita mais razão há que se indicar os bits válidos, quando o código usar um número variável de bits por caracter.

Esta observação está aqui para alertá-lo da dificuldade com o último BYTE. Não é questão trivial. Faça força para entender a jogada, e se não conseguir, não se avexe, peça ajuda.

1