# Universidade Federal de São Carlos, UFSCar, *campus* de Sorocaba

# Multimídia Computacional – Bacharelado em Ciência da Computação (2012)

# Departamento de Computação de Sorocaba – DComp

## Lista de Exercícios[1]

Observação: em todos os problemas que pedem implementações, procure desenvolver um código estruturado na forma de uma função que possa ser chamada inúmeras vezes em um programa principal. Lembre-se, esses códigos poderão ser consultados na prova e devem ser construídos de forma a facilitar a construção de programas mais complexos, assim como em funções de bibliotecas. As implementações deverão ser feitas em linguagem C ou utilizando o Octave, sendo que o contexto do exercício deixará clara qual linguagem será utilizada.

1) Um sinal de áudio contínuo pode ser representado como uma soma infinita de funções do tipo senos e cossenos. Para ilustrarmos essa ideia, considere um sinal periódico com período $T_0$ (que chamaremos de período fundamental). A medida $f_0 = \frac{1}{T_0}$ , é a frequência fundamental desse sinal (medida em Hertz – Hz , ou ciclos por unidade de tempo). Podemos ainda definir a frequência ângular dada por $w_0 = \frac{2\pi}{T_0} = 2\pi f_0$ , medida em radianos por unidade de tempo. Baseado nessas considerações, um sinal muito simples pode ser expresso como $s(t) = A * \cos(w * t)$ , onde para $A = 1$ e $w = 2$, temos um sinal que se repete duas vezes enquanto o parâmetro $t$ percorre o intervalo de comprimento $2\pi$ e com uma amplitude unitária. Observe que podemos assumir que s(t) é uma soma infinita, onde todas as outras parcelas da soma são multiplicadas por zero.

a) utilizando o Octave, visualise o sinal $s(t) = A * \cos(w * t)$ , para diferentes valores de $A$ e $w$ e observe o comportamento do sinal. Dica: construa um intervalo de amostras para representar $s(t)$ de forma conveniente;

b) visualize o sinal $s(t) = A_0 + C_1 * \cos(w * t + \theta)$ , onde $A_0$ é a altura média do sinal em relação as abscissas, e $\theta$ o ângulo de fase que corresponde ao deslocamento horizontal do sinal. Novamente, considere diferentes valores para $A_0$ e $C_1$ e verifique as diferenças;

c) desenvolvendo a expressão acima, verificamos que a função $s(t)$ pode ser expressa como $s(t) = A_0 + A_1 * \cos(w * t) + A_2 * sen(w * t)$ , onde $A_1 = C_1 * \cos(\theta)$ e $A_2 = -C_1 * sen(\theta)$ . Visualize esse sinal, variando os parâmetros e analizando seu comportamento.

2) Construa e visualize os seguintes sinais unidimencionais:

a) $s(t) = sen(t)$ ;

b) $s(t) = sen(t) + \frac{1}{3} sen(3t)$ ;

c) $s(t) = sen(t) + \frac{1}{3} sen(3t) + \frac{1}{5} sen(5t)$ . Esse sinal está se aproximando de um tipo de onda especial.Você conseguiria dizer de qual forma de onda estamos falando?

1 Muitos exercícios já foram feitos na forma de atividades em sala de aula. São repetidos aqui para enfatizar a importância dos seus conteúdos enquanto matéria de prova. Se você não fez algum completamente, faça!

d) estude os exemplos dados em aula. Em todos os casos, você deverá pensar em um número adequado de amostras para cada sinal em um determinado intervalo de tempo para a correta interpretação do exercício.

3) Faça a transformada de Fourier dos exemplos acima (pode ser no Octave). Tente enxergar as frequências utilizadas na construção dos sinais no mapa de frequências geradas pela transformada discreta de Fourier (DFT) do sinal, considerando um conjunto de amostras finito.

4) A transformada de Fourier de um sinal de áudio representa todas as frequências contidas no sinal. Utilizando o Octave, leia um sinal de áudio no formato *wave* e faça a transformada discreta de Fourier. Visualize o sinal em ambos os domínios. Posteriormente, construa sinais de áudio mais simples, com poucas frequências, assim como foi feito nos primeiros exercícios e identifique essas frequências no domínio de Fourier.

5) Repita o exercício anterior utilizando o aplicativo FFTMus2. A ideia é que você seja capaz de verificar que, de fato, a transformada de Fourier, é um mapa de frequências do sinal no domínio do tempo.

6) Considere a estrutura de dados (escrita em linguagem C) abaixo:

```
struct HeaderType { /* cabeçalho padrão de um arquivo WAVE */
  int32_t ChunkID; /* cabeçalho RIFF (Resource Interchange File Format) */
  int32_t ChunkSize; /* tamanho do arquivo menos 8 */
  int32_t Format; /* contem as letras "WAVE" (0x57415645 no formato big-endian) */
  int32_t Subchunk1ID; /* contém as letras "fmt " (0x666d7420 no formato big-endian) */
  int32_t Subchunk1Size; /* 16 for PCM */
  uint16_t AudioFormat; /* PCM = 1 (significa uma quantização linear) */
  uint16_t NumChannels; /* canais: 1 = mono e 2 = stereo */
  int32_t SampleRate; /* frequência de amostragem */
  int32_t ByteRate; /* SampleRate * NumChannels * BitsPerSample/8 */
  uint16_t BlockAlign; /* NumChannels * BitsPerSample/8 */
  uint16_t BitsPerSample; /* numero de bits por amostra: 8 ou 16 bits */
  int32_t Subchunk2ID; /* contem as letras "data" (0x64617461 no formato big-endian) */
  int32_t Subchunk2Size; /* numero de bytes para os dados */
};
/*
 * O formato de arquivo WAVE é um subconjunto da especificação RIFF da Microsoft
 * para o armazenamento de arquivos multimidia.
 */
```

Implemente um programa em linguagem C para ler um arquivo de áudio no formato *wave* e exibir na tela as especificações descritas em seu cabeçalho.

7) Dado um sinal de áudio analógico, descreva como se dá o processo de conversão para o formato digital. Você consegue descrever alguns dispositivos físicos associados a cada etapa do processo?

8) Descreva a diferença entre *amostragem* e *quantização* de um sinal contínuo.

9) Descreva o que você entende por *frequência de amostragem de um sinal* e faça uma relação com o *teorema de Shannon* e com o *limite de Nyquist*.

10) O que acontece se o *limite de Nyquist* não for respeitado? Observação: em áudio, o fenômeno de *aliasing* é frequentemente conhecido como *pseudonímia* (mas isso não responde a pergunta!).

11) Defina *bitrate*, *byterate e bitstream*. Dica, observe os comentários do cabeçalho apresentado no primeiro exercício.

12) Descreva o que significa o formado de dados PCM (*Pulse Code Modulation*) utilizado no *wave* canônico. Como é feita a digitalização dos dados (amostragem e quantização) para esse formato de armazenamento? Existe algum tipo de compressão nessa abordagem?

13) Considere um arquivo de áudio no formato *wave* onde os dados estão armazenados seguindo um critério de amostragem uniforme, sem compressão de dados, no formato PCM. Altere a frequência de amostragem do sinal, mas sem corrigir as demais informações do cabeçalho de forma a gerar um erro na reprodução do áudio. Altere a frequência de amostragem para cima e para baixo e observe o que acontece com o áudio. Explique o que você está ouvindo!

14) Considere um arquivo de áudio no formato *wave* onde os dados estão armazenados segundo um critério de amostragem uniforme, sem compressão de dados (em outras palavras, considere um arquivo de dados no formato PCM). A frequência de amostragem do áudio original deve ser de 44100 Hz. A escolha do áudio fica a critério do aluno, desde que respeite as especificações do exercício. Escreva um código em C para ler e escrever um arquivo no formato *wave*. Utilize a implementação realizada na primeira atividade prática, onde foi realizada a leitura do cabeçalho de um arquivo de áudio no formato *wave*. Após, implemente uma função para decimar o sinal por um fator definido pelo usuário. Salve o sinal de áudio original, amostrado em 44.100 Hz, decimado por fatores de 2, 4, 8, 16, 32 e 64. Para cada decimação, gere um arquivo em disco e ajuste a frequência de amostragem para refletir a decimação. A duração do áudio deve ser a mesma para todos os arquivos. Que conclusões você pode retirar sobre o processo de sub-amostragem? Observação: *decimação* significa reduzir o número de amostras do sinal. Ou seja, no exercício, corresponde a uma sub-amostragem do sinal. Por exempo, decimar um sinal de áudio por 2, significa reduzir a frequência de amostragem por 2.

15) Por qual razão arquivos de áudio *wave* em CDs comerciais, baseados no formato PCM, são amostrados considerando uma frequência de amostragem de 44.100 Hz?

16) Faça a transformada de Fourier de um sinal de áudio (utilize o Octave) amostrado a uma taxa de 44.100 Hz e mostre qual é a maior frequência contida no sinal. Você poderia estimar qual é a maior frequência apenas com uma argumentação teórica, sem precisar da transformada de Fourier?

17) Considere um arquivo de áudio no formato *wave* onde os dados estão armazenados seguindo um critério de amostragem uniforme, sem compressão de dados, no formato PCM e

a) escreva um código em linguagem C para ler um arquivo no formato *wave* e adicionar ruído

ao sinal. Salve o sinal ruidoso em outro arquivo. O ruído deve seguir uma distribuição uniforme (dica: utilize a função random);

b) dado o sinal ruidoso, implemente um código em linguagem C para minimizar o ruído baseado no critério da média das amostras vizinhas. O número de amostras utilizadas para a média deve ser um parâmetro informado pelo usuário. O que se pode concluir dessa abordagem?

c) sabemos que ruídos estão associados as altas frequências de um sinal, assim como informações de mudanças abruptas. Implemente uma função em C para minimizar ruídos, baseada em uma análise no domínio de Fourier.

18) Descreva as características *físicas* e *psicofísicas* associadas ao processamento de áudio, conforme discutido em sala de aula. Como essas características se relacionam umas com as outras? E de que forma estas são utilizadas em abordagens de compressão de dados?

19) Descreva como o formado PCM pode ser estendido para incorporar algum mecanismo de compressão de dados. Em que categoria esses formatos de compressão podem ser classificados?

20) Qual a diferença entre *codificação* e *compressão* de informação? Qual a expressão matemática utilizada para medir a taxa de compressão? Essa expressão carrega alguma medida da qualidade do áudio? Em caso negativo, qual o parâmetro mais importante para medir a qualidade de um áudio digital?

21) Como podem ser classificadas as abordagens de compressão de áudio? Para cada uma delas, qua abordagens podem ser utilizadas para esse propósito? Como essas abordagens se relacionam com as características físicas e psicofísicas do áudio?

22) Dentro das abordagens relacionadas no exercício acima, qual delas é diretamente associada ao conceito de entropia?

23) Por que algoritmos baseados em compressão com perdas permitem uma compressão de dados maior do que aqueles baseados em compressão sem perdas?

24) Conforme mencionado em sala de aula, podemos afirmar que a grande variedade dos algoritmos de compressão existentes podem ser classificados ou construídos seguindo as abordagens das respostas dos exercícios anteriores?

25) Defina o termo *codec*. Quando falamos em *codecs*, estamos nos referindo a quê especificamente?

26) Implemente uma função em C que dado um arquivo de áudio no formato *wave* canônico necessariamente com *dois canais de áudio,* o resultado seja uma compressão dos dados baseada na redundância das informações contidas nos dois canais estéreo. Gere um arquivo de saída apenas com a informação de áudio, mais a informação necessária para recuperar os dados originais.

27) Implemente uma função que, uma vez fornecido os dados gerados pelo algoritmo do

exercício anterior, gere um arquivo *wave* canônico sem compressão com dois canais de áudio.

28) Compare o resultado do algoritmo de compressão proposto acima com os dados originais do arquivo *wave* e verifique a taxa de compressão. Você considera que obteve uma boa taxa de compressão? O resultado é completamente reversível? Por quê?

29) Considerando um arquivo *wave* canônico, implemente, em linguagem C, uma função para uma versão do formato de compressão baseado no DPCM. Implemente também uma função para reverter o processo.

30) Considere um arquivo de áudio no formato *wave* onde os dados estão armazenados seguindo um critério de amostragem uniforme, sem compressão de dados, no formato PCM e

a) escreva um código em linguagem C para ler um arquivo no formato *wave* e calcule a transformada discreta do cosseno (DCT) dada pela expressão

$$F(u) = \frac{C(u)}{2} \sum_{i=0}^{N-1} \cos\left(\frac{(2i+1)\pi u}{2N}\right) f(i)$$

para u = 0,...,N-1, onde N é o tamanho do sinal, *f*(i) amostras do sinal e

$$C(\epsilon) = \begin{cases} \frac{\sqrt{2}}{2} & \text{se } \epsilon = 0 \\ 1 & \text{caso contrário} \end{cases}$$

b) avalie o resultado da aplicação da DCT sobre os dados de áudio original. Essa transformada é muito utilizada para compressão de dados, pois transfere a maior parte da informação contida para os primeiros elementos do vetor, otimizando o armazenamento (para compressão sem perdas) e facilitando a quantização dos valores (para compressão com perdas). Como o novo sinal de áudio poderia ser armazenado? Qual a taxa de compressão resultante para o sinal compactado?

c) pesquise e aplique a transformada inversa do cosseno (IDCT) – considere a expressão passada em aula – e gere um novo arquivo *wave*, no formato PCM. Os dados foram completamente recuperados? Compare com o arquivo de áudio original.

31) O algoritmo baseado da DCT (a qual é uma simplificação da DFT) é uma abordagem de compressão baseada na redundância da informação. Outros algoritmos baseados em critérios de similaridades são o *código de Huffman* e o *Run-length* (RLE), que, como veremos mais tarde, são também explorados dentro do padrão JPEG, para compressão de imagens digitais. Implemente duas funções em linguagem C que, após ler os dados armazenados em um arquivo *wave* canônico, faça a compressão dos dados utilizando o algoritmo de *Huffman* e o algoritmo RLE (implemente uma função para cada algoritmo de compressão). Implemente também duas funções para ler os dados armazenados em disco, após o uso dos algoritmos de *Huffman* e *Run-length* sobre os dados originais e que retorne arquivos *wave* sem compressão.

32) Considere um arquivo de áudio *wave*. Converta esse arquivo para um formato *mp3* (MPEG-1/2 Audio Layer 3), considerando algum *codec* específico para isso (exemplo, mpg123). Após a conversão verifique a taxa de compressão e qualidade do áudio. O que podemos concluir? Converta o arquivo no formato *mp3* novamente para o formato *wave*. Como o algoritmo de compressão *mp3* é baseado em uma abordagem com perdas podemos esperar que o arquivo *wave* convertido seja diferente do arquivo original. Verifique as distorções entre o arquivo original e o convertido a partir do formato *mp3* através da expressão do erro médio quadrático entre os dois sinais. A expressão do erro médio quadrático foi passada em aula e corresponde a

$$EMQ(s_{1,}s_2)=\frac{1}{M}\sum_{m=1}^{M}|s_1(m)-s_2(m)|^2 \quad ,$$

onde $s_1(t)$ e $s_2(t)$ são os dois sinais a serem comparados e $M$ o número de amostras dos sinais, que deve ser a mesma.

33) Implemente uma função em linguagem C para ler todos os cabeçalhos de um arquivo *mp3*, assim como as *tags* ID3v1 presentes e exibir os dados na tela do computador ou salvar os dados em um arquivo em disco.