Programação Imperativa
LCC (1^o ano)

Trabalho Prático nº 2

Ano lectivo 2009/2010

1 Objectivos e Organização

Este trabalho prático tem como principal **objectivo** a consolidação dos conhecimentos adquiridos:

- na análise e especificação de problemas;
- na escolha das estruturas de dados e respectivos algoritmos para resolver os problemas especificados;
- na codificação, na linguagem imperativa C, das estruturas de dados e dos algoritmos concebidos;
- na execução e teste dos programas C escritos;
- na documentação dos projectos de programação realizados;
- na utilização do ambiente linux para programação.

O **relatório** a elaborar, deve ser claro e muito bem estruturado; além do respectivo enunciado e da descrição de cada uma das fases executadas, deverá conter exemplos de utilização. Como foi dito, o relatório terá de ser escrito em LaTeX sugerindo-se que adoptem o esqueleto proposto que pode ser descarregado a partir da página WWW principal da disciplina, no URL:

`http://www.di.uminho.pt/~jcr/AULAS/didac/manuais/index.htm`

Para o efeito, esta folha contém vários enunciados, dos quais deverá resolver **pelo menos um** à sua escolha (para efeitos de avaliação prática da disciplina só um trabalho conta).
Cada enunciado cria um enredo completo e contém 4 pedidos que podem ir sendo satisfeitos por fases (sequencialmente).

2 Enunciados

2.1 Selecção de Projectos de I&D

Considere a seguinte situação:
Pretende-se criar, com a ajuda do computador, um sistema de informação (SI) para apoio à selecção de projectos de investigação candidatos ao financiamento disponibilizado por uma Instituição Nacional (suponha a FCT).
Cada projecto candidato é caracterizado pelos seguintes atributos:

- titulo

- instituição proponente
- nome do investigador responsável
- duração do projecto (número de anos)
- lista de palavras-chave (até 5), designando as grandes áreas/tópicos de investigação
- orçamento: gastos previstos (em contos) distribuídos pelas seguintes rubricas:
  - compra equipamento
  - pagamento tarefeiros
  - despesas com consumíveis
  - despesas com consultores
  - deslocações nacionais ou estrangeiras
  - gastos gerais
- classificação atribuída pelos 3 avaliadores; correspondente a cada um, será fornecida uma sequência de 4 letras[1] representando a nota dada a cada um dos 4 critérios de apreciação das propostas: originalidade, qualidade da proposta, idoneidade da equipe, exequibilidade.

2.1.1 aquisição dos dados

Nesta 1ª fase do trabalho, pretende-se que desenvolva um programa que lhe permita ler os dados referentes a vários projectos, *atribuir*, a cada um, um número interno de código (pode ser sequencial) e *calcular* o **total de gastos orçamentado** (financiamento solicitado) e a **classificação final** (média dos 3 avaliadores sendo, para cada um, calculada a média das notas atribuídas aos 4 parâmetros arredondada a 0 casas decimais, considerando N=1, M=2, S=3, B=4, E=5).
O programa deve escrever num ficheiro de texto, de nome dado pelo utilizador, por ordem decrescente de classificações, o **código interno**, o **título**, a **instituição**, o **financiamento solicitado**, a **classificação final** e o **subsídio a atribuir pela FCT**, calculado de acordo com o seguinte critério:

1. é 0% (o projecto é *rejeitado*), se a classificação for inferior a 3 pontos;
2. é 30% do solicitado, se a classificação for 3 pontos;
3. é 75% do solicitado, se a classificação for 4 pontos;
4. é 100% do solicitado, se a classificação for 5 pontos.

2.1.2 geração de relatórios HTML

Na segunda fase do trabalho, pretende-se que gere um conjunto de 4 páginas em HTML correspondentes às 4 listas de projectos: rejeitados, financiados a 30%, a 75% e a 100%, indicando para cada projecto o **título**, **instituição** e **investigador responsável**. Deverá, ainda, ser produzida uma página extra que servirá de índice de acesso às 4 páginas pedidas.
Esta tarefa deverá ser executada automaticamente no fim da classificação (fase anterior) quando na linha de comando de invocação, a seguir ao nome do seu programa, aparecer o *switch* `-h` seguido do nome do ficheiro.

[1] cada letra pode tomar um dos valores N, M, S, B, E, significando: Não satisfaz; Minimamente satisfatório; Satisfaz; Bom; Excelente.

2.1.3 armazenamento e pesquisa de informação

Na etape seguinte, pretende-se que guarde em memória os dados lidos referentes a cada um dos projectos, associados ao código que foi automaticamente atribuído, à classificação calculada e ao financiamento atribuído.
Para o armazenamento deve usar uma lista ligada, recorrendo a memória alocada dinamicamente.
O programa deve, depois, facultar as vulgares operações de acesso à informação, a saber:

- **listagem global** de todos os projectos, ordenada por Título do projecto;
- **procura de um projecto** dado o seu Código;
- **procura de todos os projectos** pertencentes a um dada Instituição Proponente.

2.1.4 armazenamento em ficheiro sequencial

Na quarta fase do trabalho, pretende-se que acrescente ao menu principal do seu programa uma nova opção que lhe permita guardar em memória secundária (ficheiro sequencial binário, de nome fixo) todos os dados lidos ou calculados que estão armazenados na lista ligada (criada na fase anterior).
A operação de aquisição de dados (fase 1) deve ser modificada para permitir o carregamento automático dos dados a partir do ficheiro acima, sempre que se verifique que esse ficheiro existe.

2.2 Gestão de Obras numa Editora

Considere a seguinte situação:
Pretende-se criar, com a ajuda do computador, um sistema de informação (SI) para gerir uma editora de livros e revistas, essencialmente na perspectiva da sua relação de trabalho com os autores.
O ciclo de vida de uma obra é constituído pelas seguintes fases: entrega do original (pelo autor); apreciação da obra (pelo editor); rejeição (nesse caso o processo termina aqui), ou celebração do contrato; edição (pelo revisor); revisão do texto (pelo autor); impressão.
Cada original é descrito pelos seguintes atributos:

- titulo
- nome do autor, ou autores
- número de páginas
- número de fotografias
- tipo (um entre uma lista limitada de hipóteses: romance, biografia, ficção, policial, aventura, etc.)
- ano em que foi escrito
- data de entrega
- nome do receptor (pessoa da editora que aceitou a obra para análise)
- avaliação preliminar atribuída pelo receptor; uma sequência de 4 letras (cada letra pode ser A, B, ou C) representando a primeira opinião relativa aos seguintes critérios de decisão: reconhecimento do autor; oportunidade comercial do tipo de obra; facilidade de edição/publicação; e dimensão do público alvo.

2.2.1 aquisição dos dados

Na primeira fase, pretende-se que desenvolva um programa que lhe permita ler os dados referentes aos originais recepcionados ao longo de um dia.
Depois de atribuir um número interno de código (pode ser sequencial) a cada original, deve ser calculada a sua classificação preliminar fazendo a média das notas atribuídas aos 4 parâmetros acima indicados (considere A=20, B=10, C=5).
O titulo da cada original recepcionado, juntamente com o código interno, o primeiro autor, a classificação calculada e a data provável para comunicação da decisão dos editores[2] deve ser escrito num ficheiro de texto, de nome dado pelo utilizador, ordenando as obras por autor.

2.2.2 geração de relatórios HTML

Na fase seguinte do trabalho, pretende-se que gere um conjunto de 2 páginas em HTML correspondentes às 2 listas de livros—*rejeitados* (classificação média menor do que 10 pontos), e *aceites* para apreciação (classificados com média maior ou igual a 10 pontos)—indicando para cada livro o **título** e o **nome dos autores**. As páginas apenas devem incluir livros com data de recepção menor ou igual a uma data indicada. Deverá, ainda, ser produzida uma página extra que servirá de índice de acesso às 2 páginas pedidas.
Esta tarefa deverá ser executada automaticamente no fim da classificação (fase anterior) quando na linha de comando de invocação, a seguir ao nome do seu programa, aparecer o *switch* `-h` seguido do nome do ficheiro.

2.2.3 armazenamento e pesquisa de informação

Na terceira fase do trabalho, pretende-se que guarde em memória os dados lidos referentes a cada um dos originais, associados ao código que foi automaticamente atribuído, à classificação calculada e à data provável de decisão.
Para o armazenamento deve usar uma lista ligada, recorrendo a memória alocada dinamicamente.
O programa deve, depois, facultar as vulgares operações de acesso à informação, a saber:

- **listagem global** de todos os originais registados, ordenados por Título;
- **procura de um original** dado o seu Código;
- **procura de todos os projectos** pertencentes a um dado Tipo.

2.2.4 armazenamento em ficheiro sequencial

Na etape final, pretende-se que acrescente ao menu principal do seu programa uma nova opção que permita guardar em memória secundária (ficheiro sequencial, de nome fixo) todos os dados lidos ou calculados que estão armazenados na lista ligada criada na fase anterior.
A operação de aquisição de dados (fase 1) deve ser modificada para permitir o carregamento automático dos dados a partir do ficheiro acima, sempre que se verifique que esse ficheiro existe.

2.3 Gestão da Volta a Portugal em Bicicleta

Neste projecto pretende-se criar um sistema de suporte para a conhecida Volta.

Pretende-se que esse sistema seja capaz de armazenar informação relativa a todas as etapas da prova (Note-se que vamos reduzir a quantidade de funcionalidades e informação para facilitar o desenvolvimento do trabalho).

Assim, uma prova é constituída por 10 etapas mais um prólogo, onde uma etapa é definida pelos seguintes atributos:

[2] considere que, em média, se demora 2 semanas por cada 150 páginas (faça sempre o arredondamento à semana, por excesso).

- local de partida
- local de chegada
- número de quilómetros
- tipo de etapa (normal ou contra-relógio)
- prémios da prova (vamos considerar apenas os prémios de montanha de 3 categorias)
- ...

Para que ocorra uma etapa é necessário que haja equipas e ciclistas, Assim uma equipa é definida por um nome, um país e uma lista de ciclistas; enquanto que um ciclista é definido pelos atributos:

- nome
- idade
- nacionalidade
- dorsal (número identificador único que possuem nas camisolas)
- equipa a que pertence
- tempo acumulado das etapas decorridas
- pontos acumulados das etapas decorridas
- pontos de montanha acumulados

As regras para atribuição de pontos são as seguintes:

- À chegada à meta: 1º - 25 P.; 2º - 20 P.; 3º - 16 P.; 4º - 13 P.; 5º - 10 P.; 6º - 8 P.; 7º - 6 P.; 8º - 4 P.; 9º - 2 P.; 10º - 1 P.; aos restantes é-lhes atribuído 0 pontos.
- À chegada ao prémio montanha (Categoria 1) 1º - 15 P.; 2º - 13 P.; 3º - 11 P.; 4º - 9 P.; 5º - 7 P.; 6º - 5 P.; 7º - 4 P.; 8º - 3 P.; 9º - 2 P.; 10º - 1 P.; aos restantes é-lhes atribuiío 0 ponto;
- À chegada ao prémio montanha (Categoria 2) 1º - 10 P.; 2º - 8 P.; 3º - 6 P.; 4º - 4 P.; 5º - 2 P.; 6º - 1 P.; aos restantes é-lhes atribuiío 0 pontos;
- À chegada ao prémio montanha (Categoria 3) 1º - 5 P.; 2º - 3 P.; 3º - 2 P.; 4º - 1 P.; aos restantes é-lhes atribuiío 0 pontos;

A cada etapa será depois associada uma classificação em termos de tempos dos ciclistas a passar na recta da meta, e os tempos parciais das suas passagens nos várias prémios de montanha ao longo de cada etapa.

Breve Explicação: o tempo com que os ciclistas chegam à meta final, será usado para calcular a *classificação individual*. A *classificação pontual* advém dos tempos obtidos na etapa.

Para calcular as classificações finais (individuais e pontuais), será necessário somar todos os tempos e pontos de cada etapa, para as respectivas classificações.

Para calcular a classificação de uma equipa, deve-se ter em conta o tempo total de todos os ciclistas da equipa.

O tempo com que um ciclista atinge um prémio de montanha, irá resultar na sua pontuação de montanha. Em cada prémio de montanha é feita uma classificação e, segundo as regras mostradas acima, são calculados os pontos para cada ciclista. Estes pontos são somados a todos os outros pontos de montanha obtidos nas etapas anteriores e são ainda somados a todos os outros pontos de montanha da etapa actual.

2.3.1 aquisição de dados

Na primeira fase pretende-se que desenvolva um programa que leia de um ficheiro toda a informação sobre a prova de ciclismo: etapas, equipas e ciclistas. Como opção, pode ser acrescentada as funcionalidades de (1) introduzir, a classificação temporal nas passagens nos prémios de montanha e no final da corrida, (2) alterar informação de uma equipa, (3) adicionar ou alterar informação de um ciclista, (4) alterar uma etapa (apenas se não houver uma classificação a ela associada),...

2.3.2 geração de relatórios HTML

Deve apresentar a seguinte informação em HTML

- Informação de uma prova: meta-informação + classificações (individual, pontos de montanha e equipa)
- Informação sobre cada uma das equipas: meta-informação + listagem dos ciclistas de cada equipa com a sua devida informação
- Estatísticas sobre as várias etapas: tempo médio de duração da prova, velocidade média do vencedor da prova, número de equipas nacionais e estrangeiras, número de ciclistas nacionais e estrangeiros, etc.
- ...

2.3.3 armazenamento e pesquisa de informação

Para o armazenamento de toda a informação, pretende-se que utilize listas ligadas sempre que possível.
A partir destas listas ligadas, devem ser fornecidas as seguintes funcionalidades (para além daquelas implícitas na geração de HTML):

- cálculo do camisola amarela a uma determinada etapa (atribuída ao ciclista com menor tempo individual ao fim da dada etapa)
- cálcuclo do camisola verde a uma determinada etapa (atribuída ao ciclista com mais pontos ao fim da dada etapa)
- cálculo do camisola branca a uma determinada etapa (atribuída ao ciclista com menos de 25 anos com melhor classificação individual, ao fim da dada etapa)
- procura e listagem de uma equipa
- procura e listagem de uma etapa
- procura e listagem de um ciclista
- cálculo e listagem da classificação por equipas (quer por tempo, quer por pontos) de uma etapa e de toda a prova;
- ...

2.3.4 armazenamento em ficheiro sequancial

No final a informação guardada anteriormente em lista ligada deve ser guardada em ficheiro para mais tarde poder ser carregada no sistema.

2.4 Gestão de Artigos Submetidos a uma Conferência

A principal recompensa que um investigador pode ter durante a sua carreira é a aceitação de artigos que submete a conferências, jornais, workshops, ou outros. A publicação de artigos científicos é a melhor maneira de fazer chegar a todos os cantos do globo os resultados de anos de investigação. É certo que, hoje em dia, a internet possibilita uma mais rápida disseminação de resultados de investigação. No entanto, os artigos de conferências ou jornais diferem dos conteúdos *ad-hoc* de páginas de internet, devido à existência de revisores especializados que creditam, ou não, o carácter científico e inovador da investigação relatada nos artigos.

Neste projecto pretende-se criar um sistema de submissão e gestão de artigos para conferências[3].

Deste modo é necessário guardar informação sobre as conferências em questão, e sobre os artigos que são submetidos às mesmas.

Assim, cada conferência (de uma forma simplificada) deve ser caracterizada pelos seguintes atributos:

- nome da conferencia
- hora e data limite de submissão
- lista de revisores
- percentagem de aceitação ou número máximo de artigos que serão aceites
- numero mínimo de artigos para cada revisor

Os revisores são pessoas especializadas numa ou em várias áreas de investigação e portanto serão identificados por:

- nome do revisor
- lista de áreas onde tem experiência
- ...

Os artigos, que devem ser submetidos antes da data limite de submissão da conferência, são descritos pelos seguintes parâmetros:

- numero único e sequencial (correspondente à ordem de submissão)
- título do artigo
- lista de autores
- lista de palavras chave (*keywords*)
- hora e data de submissão
- ...

Depois de submetidos, os artigos são então atribuídos aos revisores tendo em conta as suas áreas de especialização e as áreas específicas dos vários artigos. Depois de receberem os artigos, os revisores são obrigados a emitir um parecer sobre estes. O parecer não é mais do que um comentário ao trabalho descrito no artigo e uma nota que vai de -3 a 3:

- -3 : Fortemente Rejeitado
- -2 : Rejeitado

[3]Daqui em diante, utilizaremos a palavra conferência para notar qualquer um dos fins de submissão: conferência, jornal, workshop, etc.

- -1 : Fracamente Rejeitado
- 0 : Linha de Água
- 1 : Fracamente Aceite
- 2 : Aceite
- 3 : Fortemente Aceite

Daqui resulta então uma revisão que associa a um artigo, várias notas e vários comentários.

2.4.1 aquisição de dados

Numa primeira fase, deverá ser permitida a criação de conferências, ou passo a passo, ou efectuando o carregamento de várias conferências em ficheiro de texto.

Depois de se ter toda a informação sobre as conferências, é necessário fornecer a funcionalidade de submissão de artigos a uma conferência previamente escolhida. A cada artigo será atribuído um número interno (deve ser sequencial) que facilitará a identificação do mesmo.

Na altura em que a data limite se esgota, o programa deve ser capaz de atribuir (de forma automática) aos vários revisores o número mínimo de artigos, tendo em conta as restrições já indicadas. Nesta fase deve ser possível listar tanto para o ecrã como para um ficheiro as alocações de artigos aos vários revisores.

Numa nova fase, os revisores têm que ser capazes de dar o seu parecer sobre cada um dos artigos que lhes foram dados para revisão.

2.4.2 geração de relatórios HTML

Na fase segiunte do trabalho, pretende-se que seja feita a geração de páginas HTML que contenham informação completa sobre os comentários e notas dos vários revisores para cada um dos artigos revistos. Note que esta informação não deve mostrar quem foi o revisor do artigo!

No final de todo o processo de submissão e revisão dos artigos, uma opção deve ser fornecida para se poder disponibilizar, em HTML para visualização facilitada, todos os elementos finais e importantes para a conferência:

- nome da conferência
- número total de artigos submetidos
- número de papers aceites e rejeitados (para isto deve ser elaborado um método para cálculo da nota de um artigo)
- listagem de todos os artigos aceites mostrando o seu nome, os autores, a nota final, etc.
- ...

2.4.3 armazenamento e pesquisa de informação

A informação deve ser dinamicamente guardada em memória, usando-se, sempre que possível, listas ligadas.

As funcionalidades básicas que devem ser fornecidas, para além daquelas já enunciadas, são as seguintes:

- pesquisar um artigo dado o seu nome ou o seu número interno
- mostrar lista de autores, filtrando os repetidos, e mostrando quantos artigos cada um tem numa conferência.

- determinar se um artigo é candidato a *best-paper award* (se as suas notas estão entre o 2 o 3)
- listar os artigos por ordem das suas notas, e divulgar quais os que são candidatos a *best-paper* (um artigo é elegível para o prémio de melhor artigo se as suas notas estão entre o 2 o 3)
- listar os artigos sobre os quais um dado revisor irá dar ou deu o seu parecer
- ...

2.4.4 armazenamento em ficheiro sequencial

Nesta etapa final, deve ser possível guardar toda a informação final, estruturada em listas ligadas, de cada uma das conferências em memória secundária, para depois ser possível o seu carregamento.

2.5 Arquivo de letras de música com acordes de viola

Pretende-se manter um arquivo de letras de música em formato chord com colaboração distribuída.
Para tal é necessário guardar uma lista dos acordes conhecidos, e permitir ir recebendo novas músicas.
Cada música, ao ser recebida, deve levar um número identificador.
Um música vai ser definida pelos seguintes atributos:

- título
- nome dos autores
- nome do cantor ou agrupamento
- data
- nome e email de quem enviou a música
- nome do ficheiro que contem a letra da música em causa com os respectivos acordes

A Notação para inserir acordes pelo meio da letra das músicas consiste em escrever a sigla que designa o acorde pretendido entre parentesis rectos, como se exemplifica a seguir:

```
...
[Am]They made a [Am/G]record and
it [Am/F#]went in the [Am/G]charts,
The [G]sky [F]was the [C]limit.
[G] [C] [G] [G] [C] [G]
...
```

A Notação para definição de acordes é a seguinte:

```
{define: mi 0 0 1 2 2 0}
{define: la 0 2 2 2 0 x}
{define: lam 0 1 2 2 0 x}
```

2.5.1 aquisição de dados

Na primeira fase, pretende-se que

- desenvolva um programa para fazer a aquisição dos dados referentes a uma ou mais músicas, devendo indicar-se no fim quantas letras de cada autor foram lidas e qual foi o cantor/agrupamento que apareceu em mais músicas;

- acrescente a esse programa uma função que, dado o ficheiro que contém uma música (poema com acordes):
  1. Construa a lista dos acordes usados (sem repetições);
  2. Crie um ficheiro de texto só com o poema.

2.5.2 geração de páginas HTML

Nesta fase do trabalho pretende-se que gere páginas HTML para o conjunto de músicas introduzidas.
As páginas geradas deverão incluir a figura dos acordes contidos na música respectiva.
Por exemplo o acorde *la menor* definido como `{define: lam 0 1 2 2 0 x}` poderia corresponder à seguinte imagem:
Sugere-se para tal que use a biblioteca gd (experimente `locate libgd` para ver se tem essa biblioteca instalada na sua máquina). Veja o seguinte exemplo de uso da biblioteca gd:

```
#include "gd.h"
#include <stdio.h>

main(){

  gdImagePtr im;
  int white,black;
  FILE * out;

  im = gdImageCreate(140,140);
  white=gdImageColorAllocate(im,255,255,255);
  black=gdImageColorAllocate(im,0,0,0);
  gdImageLine(im,20,0,20,140,black);
  gdImageLine(im,40,0,40,140,black);
  gdImageArc(im,20,20,20,20,0,360,black);

  out=fopen("_.png","w");
  gdImagePng(im,out);
}
```

coloca em `_.png` a seguinte imagem:

2.5.3 armazenamento e pesquisa de informação

Nesta fase do trabalho, pretende-se que guarde em memória os dados lidos referentes a cada uma das músicas, associados ao código que foi automaticamente atribuido.
Para o armazenamento deve usar uma lista ligada, recorrendo a memória alocada dinamicamente.
O programa deve, depois, facultar as vulgares operações de acesso à informação, a saber:

- **listagem global** de todos as músicas, ordenada por Cantor/Agrupamento;
- **procura de uma música** dado o seu Código;
- **procura de todas as músicas** criadas numa determinada Data.

2.5.4 armazenamento em ficheiro sequencial

Na quarta fase do trabalho, pretende-se que acrescente ao menu principal do seu programa uma nova opção que lhe permita guardar em memória secundária (ficheiro sequencial, de nome fixo)

todos os dados lidos até ao momento e que estão armazenados na lista ligada criada na fase anterior.
A operação de aquisição de dados (fase 1) deve ser modificada para permitir o carregamento automático dos dados a partir do ficheiro acima, sempre que se verifique que esse ficheiro existe.

2.6 Arquivo musical em formato ABC

Pretende-se manter um arquivo de partituras na internet com colaboração distribuída.
Esse arquivo guarda partituras em formato ABC (um formato completamente textual).
Cada música, ao ser recebida, deve levar um número identificador
Um música vai ser definida pelos seguintes atributos:

- título
- nome dos autores
- nome do cantor ou agrupamento
- data
- nome e email de quem enviou a música
- nome do ficheiro que contem a partitura em formato ABC

Exemplo da notação ABC aplicada à conhecida música "Do re mi a Mimi..."

```
X: 1
K: C
M: 2/4|
L: 1/8
CDE2 | EEE2 | EFG2 | GGG | FED2 | DDD2 | EDC2
w: do re mi a mi-mi mi fa sol es-tá sol fá mi ré vai a pé mi re dó

Explicação:
  M: 2/4|             - compasso (neste caso binário)
  L: 1/8              - tempo de base de cada nota

  C=dó,D=ré,E=Mi, etc - notas com a duração do tempo de base
  G2                  - Sol com o dobro da duração do tempo de base
  |                   - barra de separação do compasso
  w: ...              - letra correspondente (separações por ' ' ou '-')
```

2.6.1 aquisição de dados

Na primeira fase, pretende-se que

- desenvolva um programa para fazer a aquisição dos dados referentes a uma ou mais músicas, devendo indicar-se no fim quantas letras de cada cantor/agrupamento foram lidas e quem foi o contribuinte que enviou mais músicas;
- acrescente a esse programa uma função que, dado o ficheiro que contém uma partitura em formato ABC:
  1. verifique se há tantas sílabas quantas as notas.
  2. supondo que a partitura só trás a primeira barra de compasso, acrescente as outras barras (todos os compasso devem ter a mesma duração).

2.6.2 geração de cancioneiros

Na última fase pretende-se gerar cancioneiros com as partituras em LaTeX e HTML.

Para gerar a imagem com a partitura comece por converter o ficheiro abc em Postscrit(ps) (usando o programa de domínio público `abc2ps` que deverá procurar na internet e instalar na sua máquina).

Seguidamente converta o ficheiro ps para Encapsulated Postscript (eps) usando o programa `ps2epsi` normalmente disponível em unix/linux. Os ficheiros eps são directamente incluíveis em LaTeX (ver exemplo de uso no texto latex original deste enunciado, disponível na página da cadeira, nomeadamente o uso de `usepackage{epsf}` e a inclusão das imagens).

Para definir quais as músicas a incluir e porque ordem, a função pedida nesta fase, aceita um ficheiro definidor que inclui linhas contendo:

- código de música ou
- nome de cantor/agrupamento (pretendendo-se incluir todas as músicas desse cantor/agrupamento)

Deverá escolher e documentar a sintaxe que deve seguir esse ficheiro.

Use o programa latex2html para geração de HTML.

2.6.3 armazenamento e pesquisa de informação

Na fase seguinte do trabalho, pretende-se que guarde em memória os dados lidos referentes a cada uma das músicas, associados ao código que foi automaticamente atribuido.
Para o armazenamento deve usar uma lista ligada, recorrendo a memória alocada dinamicamente.
O programa deve, depois, facultar as vulgares operações de acesso à informação, a saber:

- **listagem global** de todos as músicas, ordenada por Cantor/Agrupamento;
- **procura de uma música** dado o seu Código;
- **procura de todas as músicas** enviadas pela mesma Pessoa.

2.6.4 armazenamento em ficheiro sequencial

Na quarta fase do trabalho, pretende-se que acrescente ao menu principal do seu programa uma nova opção que lhe permita guardar em memória secundária (ficheiro sequencial, de nome fixo) todos os dados lidos até ao momento e que estão armazenados na lista ligada criada na fase anterior.
A operação de aquisição de dados (fase 1) deve ser modificada para permitir o carregamento automático dos dados a partir do ficheiro acima, sempre que se verifique que esse ficheiro existe.