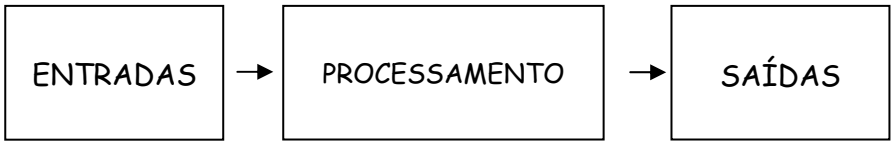
INF01202 - ALGORITMOS E PROGRAMAÇÃO TRABALHO PRÁTICO

Objetivo:

Exercitar as habilidades e conceitos de programação desenvolvidos ao longo da disciplina, através da implementação de uma aplicação em C, desenvolvida por um grupo de 1 ou 2 alunos.

O programa deve ser estruturado de forma a receber um conjunto de entradas (cuja consistência deve ser verificada), processá-las e fornecer uma ou mais saídas.



Conteúdos:

A aplicação desenvolvida deverá demonstrar os seguintes conteúdos:

1. (2 pontos) Habilidade em estruturar programas pela decomposição da tarefa em sub-tarefas, implementadas através do uso de subprogramas.
2. (2 pontos) Documentação adequada de programas (indentação, utilização de nomes de variáveis, abstração dos procedimentos para obter maior clareza, uso de comentários no código).
3. (2 ponto) Domínio na utilização de tipos de dados simples e estruturados (arranjos, estruturas) e passagem de parâmetros.
4. (1 ponto) Formatação e controle de entrada e saída, com orientação correta ao usuário.
5. (1 ponto) Utilização de arquivos.
6. (2 pontos) Atendimento aos requisitos do programa.

Aplicação a ser desenvolvida:

O trabalho a ser desenvolvido será um gerenciador de jogos para computador. O programa terá quatro funcionalidades básicas:

1 – Cadastramento de jogadores:

- Cadastramento de novos jogadores, obtendo do teclado o nome e a senha do jogador.
- Alteração de dados do jogador (campos nome e senha).
- Exclusão (pode ser lógica) de jogadores cadastrados.

2 – Cadastramento de jogos:

- Inclusão de novos jogos, obtendo do teclado o nome da pasta, nome do executável e descrição do jogo.

3 – Jogar:

- Login de jogador cadastrado, requisitando o nome de usuário (id) e senha - pré-requisito para a próxima etapa: jogar.
- Jogar, a partir da seleção de um dos jogos disponíveis.
- Atualização de resultados e estatísticas dos jogos, após a conclusão de um jogo, incluindo atualizações dos arquivos jogadores.dat, jogos.dat e ranking.txt.

4 – Consultas:

- Consulta de jogadores, informando a relação dos jogadores cadastrados e seus atributos básicos, ou, a partir de um nome ou de um id, informando resultados deste jogador.

- Consulta de jogos cadastrados, imprimindo na tela a relação e descrição dos jogos disponíveis, ou consultas específicas sobre um determinado jogo, incluindo resultados obtidos.
- Consultas sobre ranking, tanto de um jogador como de um jogo.

Estruturas de dados e arquivos utilizados:

O programa irá utilizar três arquivos binários e dois arquivos txt: jogadores.dat, jogos.dat e ranking.txt. Além disso, o programa deverá criar um arquivo texto contendo o ranking sobre essas estatísticas mediante escolha do usuário e para computar a pontuação após a execução do jogo, será usado um arquivo texto criado pelo jogo, contendo a informação.

Os arquivos e estruturas de dados correspondentes estão descritos a seguir.

1. Arquivo jogadores.dat:

```
struct  tipo_jogador
{
   int id;                                  //identificador do jogador (pode ter check-digit)
   char  nome[21];                          //nome do usuário – jogador
   char senha[9];                           //senha para fazer login
   int jogadas;                             //numero de vezes que jogou os jogos- acumulado
   int ultimo_reg_res;                      //ponteiro para último registro deste jogador em
                                            //resultados.dat
}  typedef JOGADOR;          // se quiser usar este recurso...
```

OBS: O campo ultimo_reg_est refere-se a posição do último registro deste jogador armazenado em estatisticas.dat. É inicializado com -1, indicando que o jogador ainda não jogou.

2. Arquivo jogos.dat:

```
 struct  tipo_jogo
{
      int cod;                              //código do jogo, atribuído seqüencialmente
      char jogo[21];                        //nome do jogo
      char pasta[21]                        //nome da pasta em que se encontram os dados do jogo
      char exec[21];                        //nome do executável (com extensão .exe), contido na pasta
      char descricao[255];                  //breve descrição, para fins de consulta
      int record;                           //máxima pontuação já obtida nesse jogo
      int ultimo_reg_res;                   //ponteiro para último registro deste jogo em
                                            //resultados.dat, inicializado com -1, análogo ao do jogador
} typedef JOGO;          // se quiser usar este recurso...
```

3. Arquivo resultados.dat:

```
struct tipo_resultados
{
      int id[11];                           //referencia ao id do jogador
      int cod;                              //referencia ao código do jogo
      int  pontos;                          // pontuação obtida na jogada (entre  0 e 100)
      int pos_anterior_jogador;             //ponteiro para o registro referente à jogada anterior deste
                                            // jogador (-1 indica inexistência de registro anterior)
      int pos_anterior_jogo;          //ponteiro para o registro referente à jogada anterior deste jogo
```

} typedef RESULTADOS; // se quiser usar este recurso...

OBS:. O campo pos_anterior_jogador é obtido a partir do conteúdo do campo ultimo_reg_est deste jogador em jogadores.dat, o qual é então atualizado, recebendo a posição correspondente a este registro (que está sendo incluído em resultados.dat). Da mesma forma, o campo pos_anterior_jogo é obtido a partir do conteúdo do campo ultimo_reg_est deste jogo em resultados.dat, o qual também é atualizado com a posição correspondente a este registro.

Observe que todos os jogos de um mesmo jogador podem ser obtidos, na ordem inversa em que foram jogados, a partir do acesso encadeado dos registros de um jogador (registro inicial, correspondendo ao último jogo, obtido a partir do campo *ultimo_reg_est*, seguido do registro contido em *pos_anterior*, até que o conteúdo de *pos_anterior* seja -1), o mesmo ocorrendo para todas as jogadas de um mesmo jogo.

4. Arquivo ranking.txt:

Arquivo texto, preenchido com uma lista de todos os jogos cadastrados, contendo até as cinco melhores pontuações e os jogadores que a atingiram. Esse arquivo será atualizado sempre que houver mudança nesses resultados. Abaixo temos um exemplo de organização deste arquivo para dois jogos instalados com quatro jogadores:

```
#Forca @100 &João &Maria @86 &João @73&José @65 &João @48&Ana Maria &João
&José
#Musical@76&Maria @56&João @0&João
#PacMan @
```

Note que o nome do jogo está antes da apresentação das pontuações máximas de cada jogo, antecedido por um sustenido (#). As pontuações serão antecedidas pelo símbolo arroba (@) e ordenadas da maior para a menor. Após a pontuação, deverá haver um espaço em branco e um e comercial (&) antes de cada nome, indicando o(s) jogador(es) que obteve(obtiveram) essa marca.

Todos os jogos cadastrados deverão constar no ranking. Entre um jogo e outro no arquivo deverá haver uma nova linha (LF – Line Feed). Jogos que não possuírem nenhuma pontuação deverão conter apenas um símbolo @.

OBS: Este ranking deverá ser atualizado a cada mudança nos dados, ou seja, quando um novo jogo for incluído no cadastro ou quando algum jogo for jogado e a pontuação resultante for uma das cinco melhores. Note que também é possível imprimir esse ranking na tela mediante uma opção no menu principal.

5. Arquivo out.txt:

Arquivo texto, contendo a pontuação obtida pelo jogador na execução de um jogo. Esse arquivo é gerado por cada jogo, dentro de sua pasta, podendo incluir mais de um resultado de jogo, (no caso o jogo ser jogado mais de uma vez durante uma mesma execução). O formato dos dados é o seguinte:

<pontuação1> * <pontuação2> * ... EOF

Cada pontuação será representada como uma seqüência de caracteres numéricos, que deverá ser convertida para o número inteiro correspondente à pontuação obtida. O número ficará entre zero e cem.

Jogos disponibilizados:

Serão disponibilizados dois jogos para uso no programa: Forca e Musical. Cada um deles terá sua pasta, com o respectivo nome (forca e musical), contendo o executável (forca.exe ou

musical.exe), um arquivo texto de instruções de uso do jogo (manual.txt) e arquivos auxiliares usados pelo jogo. É na pasta de cada jogo que será criado o arquivo out.txt, responsável pela pontuação do jogador.

Detalhes de Implementação:

1. Interface:

O programa deverá possuir uma interface compatível com todas as suas funcionalidades. Sugere-se a criação de um menu principal, com itens habilitados e desabilitados de acordo com a etapa de execução do programa.

Menu principal inicial:
1- Fazer Login
2- Incluir/consultar/remover jogadores
3- Incluir/consultar jogos
4- Ranking
5- Jogar

Menu principal, após Login:
1- Fazer Logout
2- Incluir/consultar/remover jogadores
3- Incluir /consultar jogos
4- Ranking
5- Jogar

Nesse exemplo, o menu inicial é basicamente um só, mas antes do login a opção Jogar permanece desabilitada (no exemplo, grafada em cinza) e, depois do login feito, substitui a inicial opção por Fazer logout. A opção Ranking dá início à atualização do arquivo correspondente, antes que os resultados sejam mostrados na tela. Os resultados mostrados podem incluir apenas um jogo ou todos os jogos cadastrados.

Abaixo, temos alguns exemplos de sub-menus, obtidos a partir do menu principal.

Menu Incluir/consultar/remover jogadores:
1- Inclui novo jogador
2- Consulta lista de jogadores cadastrados
3- Consulta jogador por nome
4- Consulta resultados do jogador
5- Exclui jogador
6- Retorna

Menu Criar/consultar/remover jogos:
1- Inclui novo jogo
2- Consulta jogos cadastrados
3- Consulta jogo
4- Retorna

Menu Jogar:
1 - <nome do jogo 1>
<descrição do jogo 1>
2- <nome do jogo 2>
<descrição do jogo 2>
...
Informe o jogo a ser executado: _

A tela de execução da opção Jogar deverá apresentar uma lista com os jogos cadastrados e esperar por um nome de jogo para ser executado.

2. Implementando a execução de um jogo:

Para executar um jogo, será necessário usar o comando system( ) da biblioteca stdlib.h. Como parâmetro para o system, pode-se utilizar qualquer comando DOS. Por exemplo, para a execução do jogo que possui nome de executável exemplo.exe e que se encontra na pasta Pexemplo, no mesmo diretório do programa, o comando ficaria:

```
system ("cd Pexemplo & exemplo.exe");
```

Por segurança, recomenda-se que, antes da execução deste comando, todos os arquivos abertos sejam fechados. Assim garante-se que, no caso de falha do jogo, não haja perda de informações dos arquivos utilizados pelo programa. Também é interessante que, após a execução deste comando, seja incluída a impressão da mensagem "jogo concluído com sucesso", seguida pela execução de um system("pause").

3. Guardando os resultados do jogo:

O acesso ao arquivo out.txt, criado pelo jogo dentro de sua pasta, é similar ao acesso aos demais arquivos texto. A única mudança é endereço do arquivo, que estará em uma pasta especial. Supondo uma pasta Pexemplo, contendo o arquivo out.txt teríamos:

```
fpointer = fopen ("Pexemplo\\out.txt", "r");
```

Observe que existem duas barras, isto porque em C uma barra apenas é usada para definições de saída. Assim, para imprimir a barra é necessário inserir duas barras.

Produto

1. Programa executável.
2. Código documentado (arquivos .c e/ou .cpp).
3. Instruções de uso: texto curto, explicando como o programa funciona, como a entrada de dados é informada ao programa, quais os recursos disponíveis, exemplos, e outras informações necessárias e suficiente para que seja executado sem o autor (Manual do Usuário).
4. Dados de teste: dados usados para testar o programa.

Funcionalidades extras:

1. Proteção da leitura da senha, colocando '*' na tela em vez de mostrar a senha quando digitada.
2. Criação/adaptação de um novo jogo, que possa ser jogado através do programa.
3. Verificação de nomes iguais de jogos ou usuários, impedindo sua inclusão.
4. Inclusão de estatísticas e métodos de consulta individuais.

Proposta: dia 15 de Outubro (quarta-feira)

Entregar em aula a composição do Grupo (duplas).

Relatório de andamento: dia 03 de Novembro (segunda-feira)

Entrega em aula de relatório escrito descrevendo as funcionalidades solicitadas para o Menu Principal, Incluir/consultar/remover jogadores e Incluir/consultar/remover jogos, com as respectivas funções de manipulação de arquivos, que equivale cerca de 50% do trabalho.

Entrega: até dia 19 de novembro (quarta-feira), incluindo:

1. Código documentado (arquivos.cpp) contendo em seu cabeçalho o nome dos alunos.
2. Programa executável
3. Instruções de uso: texto curto explicando como o funcionamento do programa, como a entrada de dados é informada ao programa, exemplos, etc.
4. Dados de teste: dados usados para testar o programa

Apresentação:

Os trabalhos serão apresentados nos dia **19 e 20 de novembro,** em aula. Somente serão apresentados programas que tenham sido entregues dentro do prazo estabelecido. Nenhuma alteração será permitida, após a entrega do trabalho.

Avaliação:
- Cada um dos conteúdos do trabalho será avaliado separadamente segundo a pontuação descrita no item **Conteúdos** acima.
- O programa deve atender todos os requisitos definidos acima, não deve apresentar erros de compilação e deve rodar corretamente.
- A nota do trabalho prático corresponderá a 50% da nota de laboratório.