Universidade de Brasília
Departamento de Ciência da Computação/IE
Semestre: 02/2013
Disciplina: Computação Básica

Trabalho Prático 4

O objetivo desta quarta etapa é que o aluno aprenda a utilizar **registros, arquivos tipo binários e tipo texto** em algoritmos e a implementá-las em programas que executem corretamente no computador.

Especificação da terceira etapa:

Problema: ALTERAR O TRABALHO 3 NOS SEGUINTES PONTOS:

1. CRIAR UM MENU COM AS SEGUINTES OPÇÕES:

    1. JOGAR
    2. CONSULTAR RANKING
    3. CRIAR NOVO TEMA
    4. SAIR

2. PERMITIR CRIAR NOVO TEMAS (arquivos .txt) E CRIAR UM ARQUIVO BINÁRIO QUE CONTÉM INFORMAÇÕES SOBRE OS TEMAS EXISTENTES:

    Ao selecionar a opção "1. JOGAR" o programa deverá verificar se existe algum tema cadastrado. Este cadastro (ou seja, os temas existentes) deve estar armazenado em um arquivo binário (chamado `temas.bin`) que conterá registros com a seguinte estrutura:

```
typedef struct {
      char nometema[20];
} tipoTema;


/* exemplo de declaração do registro */
      tipoTema Tema;
      ...
```

    Caso o programa verifique que não há temas cadastrados, ou seja, o arquivo binário está vazio (ou não existe), então o programa deve dar uma mensagem "NÃO HÁ TEMAS CADASTRADOS. PARA JOGAR É NECESSÁRIO CADASTRAR UM TEMA PRIMEIRO. ACESSE A OPÇÃO 3 DO MENU PRINCIPAL".

    Ao selecionar a opção 3 do menu principal o programa deverá permitir que o usuário cadastre um novo tema (que será o nome do arquivo texto sendo criado) e insira uma lista de palavras e a pontuação para cada palavra. O programa deverá ler para a palavra, sua pontuação e imediatamente gravá-la em uma linha do arquivo texto sendo criado. Em seguida deve ser perguntado se o usuário deseja cadastrar mais uma palavra ("DESEJA

CADASTRAR MAIS UMA PALAVRA? (S/N)"). O cadastro naquele tema (arquivo) deve prosseguir enquanto o usuário digitar "S" ou "s". Quando o usuário encerrar o cadastro de palavras, o arquivo texto sendo criado deverá ser fechado. Em seguida deve-se atualizar o arquivo binário que armazena os temas existentes. Caso este arquivo ainda não exista (no caso de ser o primeiro tema) ele deve ser criado.

IMPORTANTE: Antes de iniciar o cadastro de um novo tema, esta opção deve verificar se o tema inserido pelo usuário já existe no cadastro de temas. Lembrando que o nome (string) armazenada no arquivo binário que contém os temas existentes, será a mesma do nome físico do arquivo texto que contém as palavras daquele tema.

Para verificar se o tema já existe, o programa deve abrir o arquivo binário e pesquisar registro a registro (até o fim do arquivo), se aquele tema já existe. Caso exista, deve informar uma mensagem ao usuário ("TEMA JÁ EXISTENTE. ESCOLHA OUTRO TEMA!") e solicitar outro tema. O teste (se o tema já existe) deve ser feito sempre que o usuário informar um tema.

ATENÇÃO: não há limites para a criação de temas (arquivos texto) nem para o número de palavras cadastradas em cada tema (no arquivo texto). Pode-se assumir que cada arquivo texto contendo as palavras de cada tema foi digitado corretamente e está preenchido. Isto significa que quando vocês forem testar, deverão digitar o arquivo corretamente fazendo um cadastro válido do arquivo com as palavras e pontuações. Pode-se assumir também que a pontuação de cada palavra será digitada corretamente, isto é, não é necessário testar a consistência da pontuação.

3. SELECIONAR UM TEMA PARA JOGAR:

Ao selecionar a opção "1. JOGAR" caso seja verificado que existe(m) tema(s) cadastrado(s) (veja explicação no início do item anterior) o programa deve perguntar ao jogador um tema para iniciar o jogo. Para isso deve ser feito o seguinte:

- percorrer o arquivo binário e mostrar na tela os temas cadastrados;
- junto com cada tema deve ser mostrado um número sequencial (que inicia em 1 (um)) para que o usuário possa digitar apenas o número correspondente do tema selecionado. (veja exemplo abaixo para um caso de 5 temas cadastrados)

```
1     COMPUTACAO.TXT
2     FILMES.TXT
3     FLORES.TXT
4     GAMES.TXT
5     PROFISSOES.TXT
(DIGITE 0 (ZERO) PARA SAIR)

SELECIONE UM TEMA INFORMANDO O NÚMERO:
```

Deve ser feito um teste de validação se o número informado está correto, ou seja, não pode ser informado um número para um tema que não existe (no exemplo acima, este número não pode ser negativo, nem maior do que 5 (cinco)). Caso um número incorreto seja informado, o programa deverá dar uma mensagem adequada ("NÚMERO INVÁLIDO. DIGITE

NOVAMENTE!"). Caso o usuário digite 0 (zero) o programa deve retornar ao menu principal. Caso um número válido seja digitado o jogo deverá ser iniciado (como feito no trabalho 3) utilizando o arquivo texto daquele tema (número) selecionado.

No jogo, você deverá também incluir um comando para limpar a tela antes de mostrar a matriz da forca. Para fazer isto, inclua o seguinte trecho após os #include do seu programa:

```
#ifdef _WIN32
      #define CLEAR "cls" /* soh funciona no windows */
#else
      #define CLEAR "clear" /* soh funciona no linux ou mac (unix) */
#endif
```

Estas diretivas definem o comando que deve ser passado para a função system() (declarada no cabeçalho <stdlib.h>), para limpar a tela, de acordo com o sistema operacional que está compilando seu código. Chame a função desta forma: system (CLEAR);

4. CRIAR UM RANKING E PERMITIR CONSULTA AO RANKING

Ao selecionar a opção "1. JOGAR", depois da verificação da existência de temas e da escolha do tema (conforme explicado nos itens acima), o programa deve perguntar ao jogador o seu nome/apelido/nickname. Este nome será usado para gravar a pontuação no ranking (um arquivo binário denominado `ranking.bin`). Ao encerrar o jogo (seja por vitória ou por derrota) o programa deve abrir o arquivo binário que contém o ranking e atualizá-lo, da seguinte forma:

- caso o arquivo não exista, o programa deve criá-lo e gravar o nome do jogador e sua pontuação no arquivo.
- caso o arquivo já exista, o mesmo deve ser aberto e o nome do jogador juntamente com sua pontuação deve ser gravado no fim do arquivo.

O arquivo binário contendo o ranking conterá registros com a seguinte estrutura:

```
typedef struct {
      char nomejogador[20];
      int pontuacao;
} tipoRanking;


/* exemplo de declaração do registro */
      tipoRanking Ranking;
      ...
```

ATENÇÃO: o arquivo contendo os registros do ranking não deve estar ordenado, ou seja, os registros são gravados sempre no fim do arquivo independente da pontuação. Ao consultar o ranking o arquivo deverá ser carregado para a memória RAM onde será ordenado e mostrado na tela.

Ao selecionar a opção "1. CONSULTAR O RANKING" o programa deve limpar a tela e mostrar uma lista dos nomes dos jogadores e suas respectivas pontuações. Esta lista deve

estar ordenada em ordem decrescente de pontuação (da maior para a menor). Também deve ser mostrado ao lado de cada nome um número sequencial (que inicia em 1 (um)) representado a colocação do jogador no ranking. (veja exemplo abaixo)

```
*** RANKING ***

1     LUIZ PAULO            1750
2     KAKA                  1610
3     ESPERTINHO            1200
4     SABE TUDO              900
5     MARI                   730
6     RODRIGO                500
```

Os dados gravados no arquivo binário do ranking devem ser carregados em memória para um vetor de registros alocado dinamicamente (pois não se sabe quantos nomes estão gravados no ranking). Para saber o tamanho deste vetor de registros, utilize a mesma técnica do trabalho 3.

Depois de carregar todos os registros do arquivo do ranking para este vetor de registros, o mesmo deve ser ordenado (utilizando o algoritmo de ordenação Bubble Sort) em memória, em ordem decrescente de pontuação. Lembre-se da facilidade que você pode trocar um registro inteiro de posição dentro de um vetor, ou seja, não é necessário trocar cada campo do registro. Você irá alterar o próprio vetor de registros que está em memória, isto é, ele ficará ordenado em ordem decrescente de pontuação. Depois basta mostrar na tela conforme o exemplo acima. Obs: o número sequencial que é mostrado ao lado do nome de cada jogador, deve ser gerado na hora de mostrar na tela, ou seja, não deve ficar gravado no arquivo. Ele pode ser o próprio índice do vetor ordenado (mais 1 (um), já que o vetor começa em 0 (zero)).

Observações Gerais:

1. Incluir cabeçalho como comentário (ou seja, entre /* */), no programa fonte, de acordo com os critérios de avaliação dos trabalhos (Disponível no Moodle).
2. A data de entrega do programa é: **16/12/2013 (2a) até às 23:55 hs**, via moodle. **NÃO HAVERÁ PRORROGAÇÃO!!!**
3. O arquivo fonte deve ser compactado em um arquivo .zip ou .rar, seguindo as regras de nomenclatura especificadas na guia do aluno.
4. Sigam as determinações quanto às entradas e saídas do programa. Serão descontados pontos de programas que tiverem de ter a entrada e/ou a saída corrigidas. Se tiverem alguma dúvida quanto à formatação de entrada e de saída, entrem em contato com os monitores com antecedência em relação ao prazo de entrega do trabalho.