UTFPR - Universidade Tecnológica Federal do Paraná
DACOM - Departamento de Computação
BCC3001 :: Algoritmos e Estruturas de Dados 1

Exercícios 04 :: Arquivos binários

Instruções Gerais

- Faça todos os exercícios em um único arquivo .c. Utilize a função main() para fazer chamadas de testes às funções solicitadas pelos exercícios.
- Utilize a extensão .c, o compilador gcc e o editor de sua preferência: VS Code, Dev C++, etc.
    - Alternativamente, utilize https://replit.com/languages/c.

Lembretes:

- **Ao testar cada função na main(), lembre-se de liberar a memória manualmente alocada pela(s) função(ões), caso isso ocorra, para evitar vazamentos de memória;**
- **Feche o ponteiro do arquivo ao término da função: `fclose(file)`.**

1. Escreva uma função que grava o conteúdo de um vetor de inteiros em um arquivo. Caso o arquivo não exista, a função deve criá-lo.

   `void write_array(const char* filepath, int n, const int* v)`

   Exemplo de uso da função:

   ```
   int v[] = {1,2,3,4,5,6,7,8,9,10};
   // cria arquivo 'dados.bin' com conteúdo de 'v'
   write_array("dados.bin", 10, v);
   ```

2. Escreva uma função que lê o conteúdo de um arquivo para um vetor de inteiros, alocado em heap. A função deve retornar o endereço do vetor alocado na heap. O tamanho do vetor deve ser retornado pela função pelo parâmetro 'n'. Caso o arquivo não exista, a função deve retornar NULL.

   `int* read_array(const char* filepath, int* n)`

   Exemplo de uso da função:

   ```
   int n;
   // lê arquivo 'dados.bin' para 'v'. Preenche o tamanho de 'v' em 'n'.
   int* v = read_array("dados.bin", &n);
   print_array(n, v);
   free(v);
   ```

3. Para os exercícios que seguem, considere o tipo estruturado Movie, que deverá ser utilizado para manipular arquivos de registros de filmes:

   ```
   typedef struct {
      long id;
      char name[51];
   } Movie;
   ```

Escreva uma função que grava o conteúdo de um vetor de filmes (Movie) em um arquivo. Caso o arquivo não exista, a função deve criá-lo.

void write_all_movies(const char* filepath, int n, const Movie* v)

Exemplo de uso da função:

```
Movie list[] = {
      {6, "They Live"},
      {11,"Big Trouble in Little China"},
      {10,"The Thing"},
      {1, "In the Mouth of Madness"},
      {8, "Ghosts of Mars"},
      {23,"Halloween"},
      {7, "Village of the Damned"}
};
// cria arquivo 'dados.bin' com conteúdo de 'list'
write_all_movies("movies.bin", 7, list);
```

4. Escreva uma função que retorna a quantidade de registros de filmes contidos em um arquivo.

   int count_movies(const char* filepath)

   Exemplo de uso da função:

   ```
   // considerando o exemplo do exercício anterior
   int size = count_movies("movies.bin"); // size = 7
   ```

5. Escreva uma função que lê um registro de filme de um arquivo, na posição indicada (0, 1, 2, …). A função deve retornar uma cópia do registro. Se a posição for inválida, a função deve retornar um registro preenchido com zeros. OBS: você não deve trazer todos os registros para a RAM, em um array. A função deve ler somente somente o registro em questão.

   Movie read_movie(const char* filepath, int pos)

   Exemplo de uso da função:

   ```
   // considerando o exemplo do exercício 3
   Movie movie = read_movie("movies.bin", 4); //movie = {8,"Ghosts of
Mars"}
   ```

6. Escreva uma função que retorna a posição de um registro de filme em um arquivo, a partir de seu ID (campo Movie::id). A função deve retornar a posição (índice) que o registro ocupa no arquivo ou -1, caso ID inexistente. OBS: você não deve trazer todos os registros para a RAM, em um array. A função deve acessar registro por registro, no próprio arquivo.

   long find_movie(const char* filepath, long id)

   Exemplo de uso da função:

   ```
   // considerando o exemplo do exercício 3
   long pos = find_movie("movies.bin", 23);   // pos = 5
   ```

7. Escreva uma função que adiciona um registro ao final de um arquivo de filmes. A função deve retornar 1 (true) para sucesso ou 0 (false) em caso de erro. OBS: você não deve trazer todos os registros para a RAM, em um array.

   `int add_movie(const char* filepath, Movie movie)`

8. Escreva uma função que grava um registro em um arquivo de filmes a partir de seu ID (Movie::id), substituindo o existente. A função deve retornar 1 (true) para sucesso ou 0 (false), em caso de erro ou ID inexistente. OBS: você não deve trazer todos os registros para a RAM, em um array. A função deve acessar somente o registro especificado.

   `int update_movie(const char* filepath, Movie movie)`

9. Escreva uma função que remove um registro em um arquivo de filmes, a partir de seu ID (Movie::id). A função não apaga os dados do registro, mas apenas marca-o como removido, definindo ID = -1. A função deve retornar 1 (true) para sucesso ou 0 (false) em caso de erro. OBS: você não deve trazer todos os registros para a RAM, em um array. A função deve acessar somente o registro especificado.

   `int delete_movie(const char* filepath, long id)`

10. Escreva uma função que adiciona um registro em um registro em um arquivo de filmes, de forma ordenada crescente, a partir de seu ID (Movie::id). A função deve retornar 1 (true) para sucesso ou 0 (false) em caso de erro. Note que será necessário "reposicionar" os registros no arquivo, de forma a permitir a inserção de um novo registro na posição correta. OBS: para que o arquivo tenha todos os registros em ordem crescente de ID, é preciso inseri-los gradativamente com esta função. OBS: você não deve trazer todos os registros para a RAM, em um array. A função deve acessar registro por registro, no próprio arquivo.

    `int insert_sorted(const char* filepath, Movie movie)`

11. Considere o seguinte cenário: após a realização de várias remoções com a função **delete_movie()**, o arquivo ficará repleto de "buracos" (registros marcados com ID = -1). Escreva uma função que desfragmenta o arquivo, deixando todos os registros válidos de forma consecutiva, a partir do início do arquivo. OBS: você não deve trazer todos os registros para a RAM, em um array. A função deve acessar registro por registro, no próprio arquivo. A função deve retornar 1 (true) para sucesso ou 0 (false) em caso de erro.

    `int defrag(const char* filepath)`