UFRGS
UNIVERSIDADE FEDERAL
DO RIO GRANDE DO SUL

INF101202
Algoritmos e Programação
Modalidade Ead – Turma H

Material de apoio: arquivos – parte 4
Arquivos binários - acesso randômico

.inf INSTITUTO DE INFORMÁTICA UFRGS

Material original desenvolvido por
Mara Abel.
e
Cora H. F. P. Ribeiro
Revisado e complementado por
Maria Aparecida Castro Livi
e
Magda Bercht

.inf INSTITUTO DE INFORMÁTICA UFRGS

Arquivos binários acesso randômico

Acesso randômico

O que significa acesso randômico?

Acesso a pontos determinados de um arquivo, sem ordem determinada.

A posição desejada de leitura ou escrita pode não ser coincidente com a posição corrente do arquivo válida em um certo momento.

Acesso randômico

Como alterar a posição corrente de acesso a um arquivo sem varredura sequencial do mesmo?

Alterando o posicionamento sobre o arquivo com uso da função fseek, previamente a uma ação de leitura ou escrita.

A função fseek define uma nova posição corrente em um arquivo com base em:

- um ponto inicial (origem) que pode ser:
  - o ínicio do arquivo;
  - o ponto corrente (atual);
  - o fim do arquivo;
- mais um deslocamento expresso em número de *bytes*.

Função fseek: altera posição corrente de um arquivo

fseek(FILE *,numbytesdesloc,origem)

Move a posição corrente de leitura/escrita para a posição calculada como origem+numbytesdesloc
Os valores possíveis, definidos, para origem são:
SEEK_SET ou 0 = Início do arquivo
SEEK_CUR ou 1 = Ponto corrente no arquivo
SEEK_END ou 2 = Fim do arquivo

ATENÇÃO: fseek SÓ POSICIONA, NÃO LÊ NEM GRAVA !

```
...
  FILE *arq;
  struct estrutura buffer;
  if  (arq = fopen(nome,"r+b"))
     {
       fseek(arq,1*sizeof(struct estrutura),SEEK_SET);
      /*posicao corrente no inicio da segunda estrutura no arquivo*/
      if (fread(&buffer,...,...,arq) == 1)
      ...
  ...
```

Função ftell: informa a posição corrente de um arquivo

ftell(FILE *arq)

⇨ **Indica** a posição corrente no arquivo
Posição dada em *bytes*

```
...
  printf("\nposicao atual em bytes antes leitura%d\n", ftell(arq));
  if (fread(&buffer,sizeof(struct atleta),1,arq) == 1)
      { //le e confirma se o que foi lido eh estrutura, entao imprime
         printf("\nposicao atual em bytes pos leitura%d\n", ftell(arq));
...
```

Uma questão importante: o quê acontece quando se lê ou escreve sobre um arquivo

Ocorre um deslocamento sobre o mesmo.

Na leitura, lê-se um certo número de *bytes* do arquivo, assim ao concluir-se a mesma, o posicionamento sobre o arquivo estará alterado em relação ao momento anterior à leitura de um valor igual ao número de *bytes* lidos.

Na escrita, grava-se um certo número de bytes sobre o arquivo, assim igualmente ao concluir-se uma ação de escrita avançou-se sobre o arquivo.

Alteração da idade de um atleta determinado, em um arquivo gerado de forma sequencial

Na apresentação anterior, um arquivo com dados de atletas foi gerado de forma sequencial.

A seguir será apresentado um código em que esse arquivo tem o campo idade de um atleta determinado (o nome do atleta é fornecido via teclado) alterado.

Como o arquivo foi gerado de forma sequencial, e é desconhecida a posição onde se encontram os dados do atleta pesquisado, o arquivo tem que ser lido sequencialmente, desde o início, até localizar-se a posição da alteração, comparando-se o nome do atleta que se busca alterar, com os nomes dos atletas armazenados no arquivo.

Alteração da idade de um atleta determinado, em um arquivo gerado de forma sequencial

```
/*
  Altera um arquivo com dados de uma estrutura
  com informacoes de atletas.
*/
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#define MAXNOME 30
FILE *arq;
struct atleta
    {
       char nome[MAXNOME];
       int idade;
       float altura;
    };
void AlteraIdade(FILE *, struct atleta, int);
FILE * AbreArquivo(int , char *);
int main( )
  {
    struct atleta buffer; // buffer aqui eh variavel
    system ("color 70");
    AlteraIdade (arq, buffer, MAXNOME);
    system("pause");
    return 0;
  }
```

Alteração da idade de um atleta determinado, em um arquivo gerado de forma sequencial

```
void AlteraIdade(FILE *arq, struct atleta buffer, int max)
  {
     int encontrado = 0, atletascont = 0;
     char procurado[max];
     if(!(arq = AbreArquivo(max, "r+b")))
        {
              printf("Erro abertura");
              system("pause");
        }
     else
       (...)
```

A continuação do código tem a varredura sequencial do arquivo.

A cada leitura é verificado se o atleta procurado foi encontrado.

cont.

Alteração da idade de um atleta determinado, em um arquivo gerado de forma sequencial

```
...
  {
    fflush(stdin);
    printf("Nome do procurado: ");
    fgets(procurado, sizeof(procurado), stdin);
    if (procurado[strlen(procurado) - 1]
       procurado[strlen(procurado) - 1]
    fflush(stdin);
    while(!feof(arq))
    if (fread(&buffer,sizeof(struct atleta),1,arq) == 1)
        {
          if (!(strcmp(buffer.nome,procurado)))
              {
                printf("\nIdade cadastrada %d",buffer.idade);
                printf("\nNova Idade: ");
                scanf("%d",&buffer.idade);
                fseek(arq,atletascont*sizeof(struct atleta),SEEK_SET);
                              /*posiciona no ponto para substituir estrutura*/
                fwrite(&buffer,sizeof(struct atleta),1,arq); /*substitui*/
                fflush(arq); /*descarrega p/ estrutura antiga ser apagada*/
                encontrado = 1;
              }
            atletascont++; //indica em que ocorrência da estrutura do arquivo se está no momento
        }

    if (!encontrado)
        printf("\nNenhum atleta encontrado");
    fclose(arq);
  }
}
```

Aqui o acesso aos dados é sequencial

Aqui o acesso aos dados é randômico

cont.

Alteração da idade de um atleta determinado, em um arquivo gerado de forma sequencial

Ainda um trecho do código anterior

```
while(!feof(arq))
   if (fread(&buffer,sizeof(struct atleta),1,arq) == 1)
      {
        if (!(strcmp(buffer.nome,procurado)))
           {
             printf("\nIdade cadastrada %d",buffer.idade);
             printf("\nNova Idade: ");
             scanf("%d",&buffer.idade);
             fseek(arq,atletascont*sizeof(struct atleta),SEEK_SET);
                          /*posiciona no ponto para substituir estrutura*/
             fwrite(&buffer,sizeof(struct atleta),1,arq); /*substitui*/
             fflush(arq); /*descarrega p/ estrutura antiga ser apagada*/
             encontrado = 1;
           }
        atletascont++; //indica em que ocorrência da estrutura do arquivo se está no momento
      }
```

Quando o atleta procurado for encontrado, será necessário voltar a apontar para a posição do arquivo onde estão os dados a alterar (uso da função *seek*), pois cada leitura altera o ponto de acesso no arquivo.

Em procedimentos de alteração, é interessante apresentar os valores existentes no arquivo anteriores à modificação, antes de efetivar a operação.

cont.

Alteração da idade de um atleta determinado, em um arquivo gerado de forma sequencial

```
FILE * AbreArquivo(int max, char *modo)
{
     char nomearq[max];
     printf("Nome do arquivo (com no maximo %d caracteres): ", max - 1);
     fflush(stdin);
     fgets(nomearq, sizeof(nomearq), stdin);
     if (nomearq[strlen(nomearq) - 1] == '\n')
       nomearq[strlen(nomearq) - 1] = '\0';
     return fopen(nomearq,modo);
}
```

Criação e listagem de arquivo de forma randômica

Os três códigos a seguir apresentam a criação e listagem de um arquivo com informações de atletas (armazenadas em uma estrutura).

A criação do arquivo é de forma randômica, ou seja, há uma relação entre os dados armazenados e as posições que esses dados ocupam no arquivo. No exemplo, os dados de um atleta são armazenados no arquivo na posição correspondente a seu código – 1, considerando-se que a primeira estrutura do arquivo está em uma posição zero do mesmo.

Duas formas de listagem do arquivo são apresentadas:

- uma que apresenta todos as posições do arquivo, quer contenham dados fornecidos pelo usuário quer não;
- outra que apresenta apenas as posições que receberam dados.

Criação randômica do arquivo com dados de atletas: código

```
/*
  Cria um arquivo com dados de uma estrutura
  com informacoes de atletas, de forma randômica.
*/
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#define MAXCOD 10
#define MAXNOME 21
FILE *arq;
struct atleta
    {
       char nome[30];
       int idade;
       float altura;
    };

...
```

Criação randômica do arquivo com dados de atletas: código

```
int main( )
  {
     int encontrado = 0, contgrav = 0, cod, op;
     struct atleta buffer;
     char nome[MAXNOME], procurado[30];
     system ("color 70");
     fflush(stdin);
     printf ("Nome do arquivo (com no maximo %d caracteres): ", MAXNOME - 1);
     fgets(nome, sizeof(nome), stdin);
     if (nome[strlen(nome) - 1] == '\n')
        nome[strlen(nome) - 1] = '\0';
     fflush(stdin);
     if(!(arq = fopen(nome,"wb")))
        { /*abre para leitura/escrita*/
              printf("Erro abertura");
              system("pause");
        }
     else
        {
         /* TRECHO DE CRIAÇÃO DO ARQUIVO NO SLIDE A SEGUIR*/
         fclose(arq);
        }
    system("pause");
    return 0;
  }
```

cont.

Criação randômica do arquivo com dados de atletas: código

```
...
   do
   {
         do
           {
              printf("Codigo do atleta: ");
              scanf("%d", &cod);
              if (cod < 1 || cod > MAXCOD)
                 printf("\nCodigo deve estar entre 1 e %d!\n", MAXCOD);
           }
         while (cod < 1 || cod > MAXCOD);
         cod = cod - 1;
         fflush(stdin);
         printf("\nNome: ");
         fgets(buffer.nome, sizeof(buffer.nome), stdin);
         if (buffer.nome[strlen(buffer.nome) - 1] == '\n')
            buffer.nome[strlen(buffer.nome) - 1] = '\0';
         fflush(stdin);
         printf("\nIdade: ");
         scanf("%d",&buffer.idade);
         printf("\nAltura: ");
         scanf("%f",&buffer.altura);
         //fseek(arq,cod*sizeof(struct atleta),SEEK_SET);
         fseek(arq,cod*sizeof(struct atleta),0); // esta linha faz o mesmo que a anterior,
                                                 //no final, SEEK_SET ou 0 é o mesmo.
         fwrite(&buffer,sizeof(struct atleta),1,arq);
         fflush(arq);
         printf("\n1-InserirNovo, 2-Encerrar\n");
         scanf("%d", &op);
      }
   while(op != 2);
```

cont.

Criação randômica do arquivo com dados de atletas: execução

```
D:\Aparecida\LINGUAGEMC20081\CriaAtletaRand.exe
Nome do arquivo: c:\atleta.dat
Codigo do atleta: 10

Nome: Zaira Antunes

Idade: 23

Altura: 1.98

1-InserirNovo, 2-Encerrar
1
Codigo do atleta: 1

Nome: Ana Sousa

Idade: 22

Altura: 2

1-InserirNovo, 2-Encerrar
1
Codigo do atleta: 9

Nome: Pedro Meira

Idade: 21

Altura: 2.1

1-InserirNovo, 2-Encerrar
2
Pressione qualquer tecla para continuar. . . _
```

Como acontece a criação randômica de um arquivo binário

Se as posições forem criadas com intervalos entre si (no exemplo anterior foram inseridos dados do atleta de código 10, 1 e 9), o sistema criará as posições intermediárias e inicializará os campos dessas com o valor 0 ou semelhante.

A leitura acontecerá sem problemas, tanto das posições geradas explicitamente pelo usuário, quanto pelas posições intermediárias geradas pelo sistema, que estejam dentro dos limites do arquivo.

Por isso, ou verifica-se se a posição acessada em um arquivo contém efetivamente dados válidos ou corre-se o risco de apresentar informação inválida ou, pelo menos, sem interesse (ver o resultado da execução do próximo programa).

Listagem sequencial do arquivo com dados de atletas gerado de modo randômico, apresentando todas as posições, mesmo aquelas para as quais não foram fornecidos dados: código

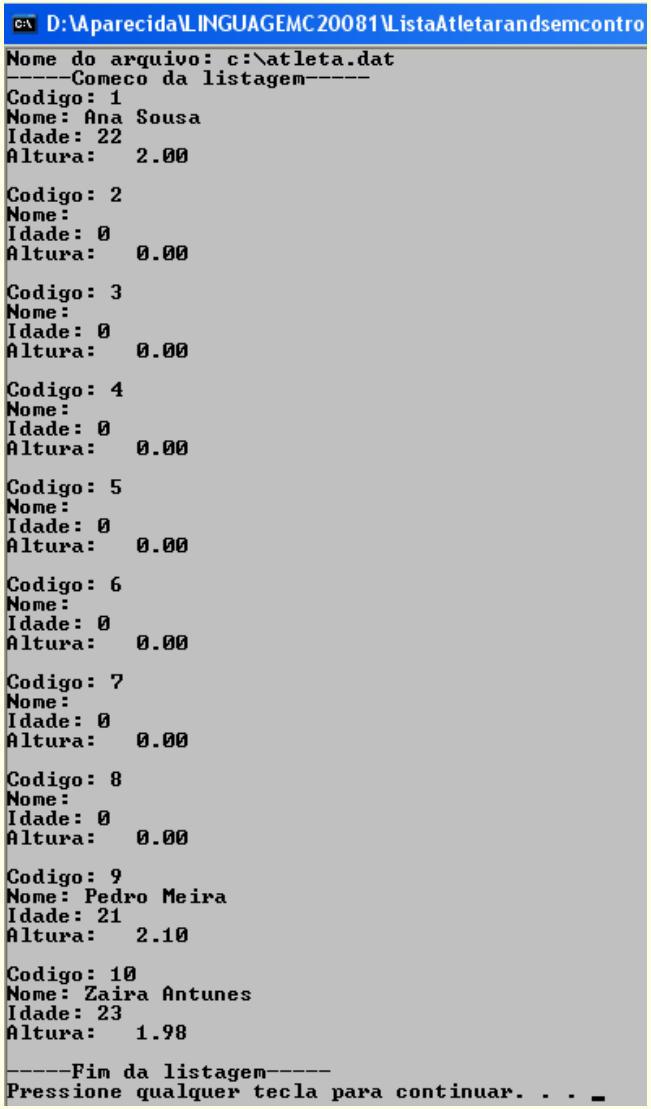
```
/*
  Lista sequencialmente um arquivo com dados de uma estrutura
  com informacoes de atletas. Arquivo gerado randomicamente.
  Na listagem nao ha controle para bloquear a apresentacao de posicoes
  para as quais nao foram fornecidos dados.
*/
#include <stdio.h>
#include <stdlib.h>
FILE *arq;
struct atleta
    {
       char nome[30];
       int idade;
       float altura;
    };
int main( )
  {
     struct atleta buffer;
     int contlidos = 0;
     char nome[15];
     int op;
     system("color 70");
     printf("Nome do arquivo: ");
     gets(nome);
     fflush(stdin);
     ...
```

cont.

Listagem sequencial do arquivo com dados de atletas gerado de modo randômico, apresentando todas as posições, mesmo aquelas para as quais não foram fornecidos dados: código

```
...
  if(!(arq = fopen(nome,"rb")))
    {
      printf("Erro abertura");
      system("pause");
    }
  else
    {
      printf("-----Comeco da listagem-----\n");
      while(!feof(arq))
          if (fread(&buffer,sizeof(struct atleta),1,arq) == 1)
              {//lê e confirma se o que foi lido é estrutura, então imprime
                contlidos++;
                printf("Codigo: %d\n", contlidos);
                printf("Nome: %s\n",buffer.nome);
                printf("Idade: %d\n",buffer.idade);
                printf("Altura: %6.2f\n\n",buffer.altura);
              }
      printf("-----Fim da listagem-----\n");
      fclose(arq);
      system("pause");
      return 0;
    }
}
```

Listagem sequencial do arquivo com dados de atletas gerado de modo randômico, apresentando todas as posições, mesmo aquelas para as quais não foram fornecidos dados: execução



Listagem sequencial do arquivo com dados de atletas gerado de modo randômico, apresentando só posições para as quais foram fornecidos dados: código

```
//até aqui igual ao código anterior
 printf("-----Comeco da listagem-----\n");
   while(!feof(arq))
       if (fread(&buffer,sizeof(struct atleta),1,arq) == 1)
           {//lê e confirma se o que foi lido é estrutura, então imprime
             contlidos++;
             if (buffer.idade > 0)
                {
                        printf("Codigo: %d\n", contlidos);
                        printf("Nome: %s\n",buffer.nome);
                        printf("Idade: %d\n",buffer.idade);
                        printf("Altura: %6.2f\n\n",buffer.altura);
                }
           }
   printf("-----Fim da listagem-----\n");
//após igual ao código anterior
```

Listagem sequencial do arquivo com dados de atletas gerado de modo randômico, apresentando só posições para as quais foram fornecidos dados: execução

```
D:\Aparecida\LINGUAGEMC20081\ListaAtletarand.exe
Nome do arquivo: c:\atleta.dat
-----Comeco da listagem-----
Codigo: 1
Nome: Ana Sousa
Idade: 22
Altura:   2.00

Codigo: 9
Nome: Pedro Meira
Idade: 21
Altura:   2.10

Codigo: 10
Nome: Zaira Antunes
Idade: 23
Altura:   1.98

-----Fim da listagem-----
Pressione qualquer tecla para continuar. . .
```

Operações básicas sobre arquivos

- inclusão
- exclusão
- alteração
- listagem
- consulta

Erros que devem ser verificados nas operações básicas sobre arquivos.

Inclusão: inclusão para já existente.

Exclusão: exclusão para inexistente.

Alteração: alteração para inexistente.

Consulta: consulta para inexistente.

Dado inexistente em um arquivo:
o quê significa?

Ou

a posição procurada está entre as posições existentes no arquivo, mas não contém dados válidos: porque foi gerada pelo sistema, ou porque foi logicamente excluída;

ou

a posição procurada está além das posições existentes no arquivo.

Exclusão lógica

O quê significa exclusão lógica?

Excluir logicamente um elemento de dados de um arquivo, significa marcar esse elemento de dados de forma a identificá-lo como não mais pertencente ao conjunto de dados válidos do arquivo. Os dados permanecem fisicamente no arquivo, mas não mais são considerados como integrando o conjunto de dados válidos. Há aplicações em que as posições excluídas são reutilizadas e outras aplicações onde isso não ocorre.

A marca de exclusão pode ser colocada como um valor especial em um campo de dados já existente, ou pode ser definido no arquivo um campo específico para registrar essa situação.

Exclusão lógica randômica: etapas de verificação e execução

Etapas de uma exclusão lógica randômica:

1 acessar a posição do arquivo para verificar se ela existe (fseek + fread):

- se existe, seguir com a exclusão (ir para a etapa 2);
- se não existe, acusar erro e parar com a exclusão.

2 verificar se a posição está preenchida com dados válidos. Posição com dados não válidos: posição gerada pelo sistema (testar algum campo contra valores conhecidos), posição excluída (verificar marca de exclusão):

- se não: acusar erro e parar a exclusão;
- se sim: seguir com a exclusão (ir para a etapa 3).

3 Confirmar se o usuário deseja realizar a exclusão:

- se não: parar a exclusão;
- se sim: seguir com a exclusão (ir para a etapa 4).

4 Reposicionar-se no arquivo (fseek), uma vez que o acesso para verificar se a posição existia fez com que houvesse um deslocamento sobre o arquivo.

5 Regravar os dados na posição do arquivo a excluir, com a marca de exclusão.

Alteração randômica: etapas de verificação e execução

Etapas de uma alteração randômica:

1 acessar a posição do arquivo para verificar se ela existe (fseek + fread):

se existe, seguir com a alteração (ir para a etapa 2);

se não existe, acusar erro e parar com a alteração.

2 verificar se a posição está preenchida com dados válidos. Posição com dados não válidos: posição gerada pelo sistema (testar algum campo contra valores conhecidos), posição excluída (verificar marca de exclusão):

- se não: acusar erro e parar a alteração;
- se sim: seguir com a alteração (ir para a etapa 3).

3 Solicitar os dados e preparar-se para a gravação.

4 Reposicionar-se no arquivo (fseek), uma vez que o acesso para verificar se a posição existia fez com que houvesse um deslocamento sobre o arquivo.

5 Gravar a alteração no arquivo.

Se for desejado apresentar as informações pós alteração, direto do arquivo:

refazer o passo 4 (fseek) e executar nova leitura (fread).

Inclusão randômica: etapas de verificação e execução

Etapas de uma inclusão randômica:

1 acessar a posição do arquivo para verificar se ela existe (fseek + fread):

se não existe,seguir com a inclusão (ir para a etapa 2).

se existe:

verificar se é posição gerada pelo sistema (testar algum campo contra valores conhecidos):

- se sim: seguir com a inclusão (ir para a etapa 2).
- se não: verificar se é posição excluída.

Se a aplicação permite reutilizar posições excluídas:

- seguir com a inclusão (ir para a etapa 2);

Se a aplicação não permite reutilizar posições excluídas:

- Acusar erro e parar com a inclusão.

2 Solicitar os dados e preparar a variável para a escrita.

3 Reposicionar-se no arquivo (fseek), uma vez que o acesso para verificar se a posição existia fez com que houvesse um deslocamento sobre o arquivo.

4 Gravar a alteração no arquivo.

Alteração de um arquivo com dados de atletas

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#define MAXCOD 10
#define MAXNOME 31
FILE *arq;
struct atleta
    {
       char nome[MAXNOME];
       int idade;
       float altura;
    };
int AlteraDados(FILE *, struct atleta, int);
void MostraAtleta (struct atleta , char *, int);
void RecebeDadosDaAlteracao (struct atleta *, int);
```

cont.

Alteração de um arquivo com dados de atletas

```
int main( )
  {
    int op, codigo, resultalt;
    struct atleta buffer;
    system ("color 70");
    char nome[MAXNOME];
    fflush(stdin);
    printf
    ("Nome do arquivo (com no maximo %d caracteres): ", MAXNOME - 1);
    fgets(nome, sizeof(nome), stdin);
    nome[strlen(nome) - 1] = '\0';
    fflush(stdin);
    if(!(arq = fopen(nome,"r+b")))
        { /*abre para leitura/escrita*/
              printf("Erro abertura");
              system("pause");
        }
    else
        {
          ...
```

cont.

Alteração de um arquivo com dados de atletas

```
        do
         {
           do
            {
              printf("\n**Alteracao de dado de cliente**\n");
              printf("\nCodigo cliente: \n");
              scanf("%d" , &codigo);
              if (codigo < 1 || codigo > MAXCOD)
                 printf
                  ("\nCodigo invalido deve estar entre 1 e %d\n", MAXCOD);
            }
           while (codigo < 1 || codigo > MAXCOD);
           resultalt =  AlteraDados (arq, buffer, codigo) ;
           if (resultalt == 1)
              printf("\nAlteracao para inexistente - codigo %d\n", codigo);
              if (resultalt == 2)
                  printf("\nAlteracao cancelada - codigo %d\n", codigo);
           printf("\n1 - AlteraOutroCliente;  0 -  Para\n");
           scanf("%d", &op);
         }
        while (op);
        fclose (arq);
        system("pause");
        return 0;
       }
}
```

cont.

Alteração de um arquivo com dados de atletas

```
int AlteraDados(FILE *arq, struct atleta buffer, int cod)
  {
     int  opalt;
     char titul[MAXNOME];
     cod--;
     fseek(arq,cod*sizeof(struct atleta),SEEK_SET);
     if (fread(&buffer,sizeof(struct atleta),1,arq) == 1)
        {
          if (buffer.idade)
              {
                strcpy(titul, "Antes da Alteracao");
                MostraAtleta (buffer , titul,  cod);
                do
                 {
                  printf
                   ("\nPara alterar: nome - 1; idade - 2\n");
                  printf
           ("\nPara alterar: altura - 3, todos - 4, cancelar alteracao - 0\n");
                  scanf("%d", &opalt);
                  if (opalt < 0 || opalt > 4)
                     printf("\nOpcacao invalida - deve ser entre 1 e 4!\n");
                  }
                while (opalt < 0 || opalt > 4);
```

cont.

Alteração de um arquivo com dados de atletas

```
            if (opalt)
               {
                RecebeDadosDaAlteracao (&buffer, opalt);
                fseek(arq,cod*sizeof(struct atleta),SEEK_SET); /*posiciona*/
                fwrite(&buffer,sizeof(struct atleta),1,arq); /*substitui*/
                fflush(arq); /*descarrega p/ estrutura antiga ser apagada*/
                printf
                  ("\n>>>>Alteracao do atleta %d efetuada!\n\n", cod + 1);
                fseek(arq,cod*sizeof(struct atleta),SEEK_SET);
                                                        /*reposiciona*/
                fread(&buffer,sizeof(struct atleta),1,arq); /*le o alterado*/
                strcpy(titul, "Depois da Alteracao");
                MostraAtleta (buffer , titul,  cod);
                return 0; // alteracao ok
               }
            else
                return 2; // alteracao cancelada
          }
       else
           return 1;
             // alteracao para inexistente (posicao existe mas sem dados)
       }
   else
      return 1; //alteracao para inexistente (posicao nao existe)
}
```

cont.

Alteração de um arquivo com dados de atletas

```
void MostraAtleta (struct atleta buffer, char *titulo, int cod)
{     printf("\n***%s***\n\n", titulo);
      printf("Codigo: %d\n", cod + 1);
      printf("Nome: %s\n",buffer.nome);
      printf("Idade: %d\n",buffer.idade);
      printf("Altura: %6.2f\n\n",buffer.altura);
      printf("\n\n");
 }
```

cont.

Alteração de um arquivo com dados de atletas

```
void RecebeDadosDaAlteracao (struct atleta *buffer, int op)
{
   switch (op)
     {
       case  1:  printf("\nNovo Nome: ");
                 fflush(stdin);
                 fgets((*buffer).nome, sizeof((*buffer).nome), stdin);
                 if ((*buffer).nome[strlen((*buffer). nome) - 1] == '\n')
                      (*buffer).nome[strlen((*buffer).nome) - 1] = '\0';
                 fflush(stdin);
                 break;
       case  2:  printf("\nNova Idade: ");
                 scanf("%d",&(*buffer).idade);
                 break;
       case  3:  printf("\nNova Altura: ");
                 scanf("%f",&(*buffer).altura);
                 break;
       case  4:  printf("\nNovo Nome: ");
                 fflush(stdin);
                 fgets((*buffer).nome, sizeof((*buffer).nome), stdin);
                 if ((*buffer).nome[strlen((*buffer). nome) - 1] == '\n')
                      (*buffer).nome[strlen((*buffer).nome) - 1] = '\0';
                 fflush(stdin);
                 printf("\nNova Idade: ");
                 scanf("%d",&(*buffer).idade);
                 printf("\nNova Altura: ");
                 scanf("%f",&(*buffer).altura);
                 break;
     }
  }
```

Outras funções de arquivo

- **rewind**
- **rename**
- **remove**

Função rewind: posição corrente para o início

rewind(FILE *)

➡ **Seta** a posição corrente do arquivo para o início

```
...
  FILE *arq;
  arq = fopen(...);
  ...
  rewind(arq);
  /*posição corrente = início*/
  ...
...
```

Função rename: renomeia ou move arquivo

rename(NomeAntigo,NovoNome)

Renomeia um arquivo ou o **Move** para um diretório destino
Obs.: o arquivo deve estar fechado!

```
...
  ...
  rename("a.dat","b.dat");
  /* a.dat -> b.dat */
  ...
...
```

Função remove: apaga arquivo

remove(NomeDoArquivo)

➡ **Apaga** o arquivo especificado do disco
Obs.: o arquivo deve estar fechado!

```
...
  ...
  remove("tourstart.exe");
  /*remove o arquivo especificado*/
  ...
...
```