

INF01202

Algoritmos e Programação

Modalidade Ead – Turma H

Material de apoio: Arquivos binários - acesso sequencial

**Material original desenvolvido por
Mara Abel.**

e

**Cora H. F. P. Ribeiro
Revisado e complementado por
Maria Aparecida Castro Livi**

e

Magda Bercht

Tipos de arquivos em C

✓ Existem 2 tipos de arquivo em C:

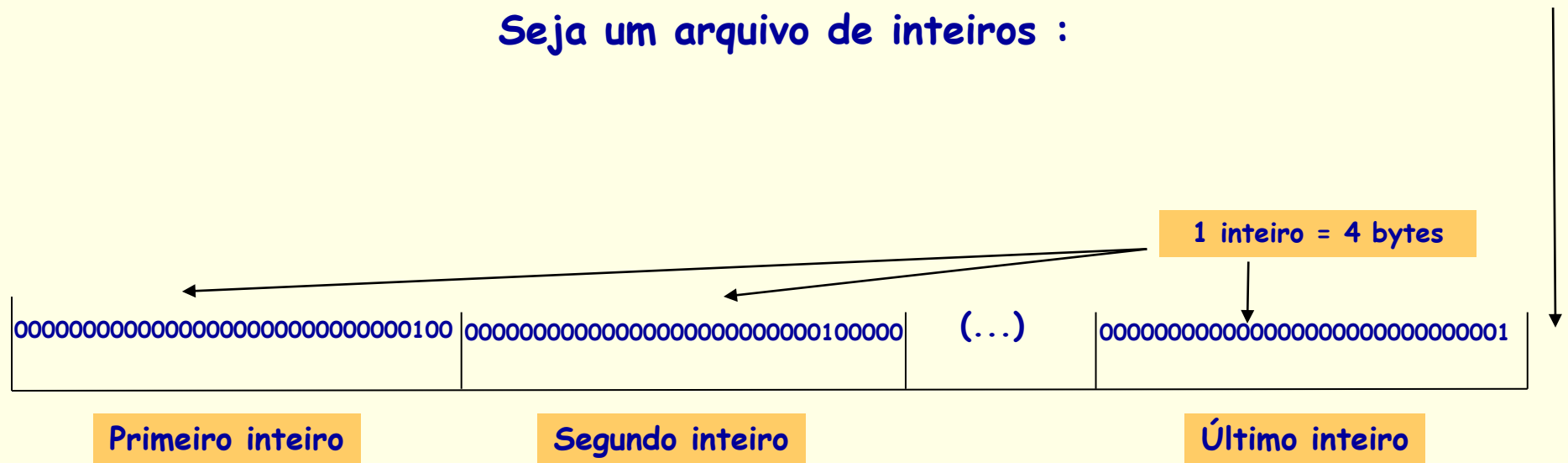
- ➔ **Binário:** contém uma sequência de bytes que apresenta uma correspondência de um para um com os bytes recebidos do dispositivo externo. Não há conversão ou tradução do que é lido no momento do armazenamento.
- **Texto:** não existe necessariamente uma correspondência de um para um entre os caracteres lidos e os armazenados. Um caractere de nova linha, por exemplo, pode ser armazenado como dois, um de retorno de carro e outro de alimentação de linha, conforme o sistema.

Arquivos: armazenamento

As separações entre *bytes* são apenas ilustrativas, nos dispositivos de memória auxiliar não existem!

Fim do
arquivo

Seja um arquivo de inteiros :



Operações básicas sobre arquivos: lembrando

Abertura

antes de acessar um arquivo, ele tem que ser aberto.

O modo de abertura define as ações que podem ser realizadas sobre o arquivo: só leitura, só escrita, leitura e escrita.

Na abertura o sistema associa um elemento do programa - o ponteiro para o arquivo - com o arquivo existente fisicamente no sistema de arquivos. Igualmente nesse momento o sistema aloca em memória as áreas intermediárias (*buffers*) para trabalhar com o arquivo.

Fechamento

ao encerrar o programa, normal ou anormalmente, os arquivos são fechados. É nesse momento que as áreas intermediárias (*buffers*) são descarregados obrigatoriamente. Se o fim for anormal, esse processo pode não acontecer de modo adequado e informações podem ser perdidas.

5

Número de arquivos abertos ao mesmo tempo:
regra geral há um número limite por implementação.

Arquivo Binário

- agregado de *bytes*;
- cada elemento pode ser um dado simples ou estruturado;
- não tem limitação de tamanho;
- **comprimento do arquivo**: número de elementos que apresenta no momento;
- elementos não têm nome;
- somente um elemento é acessível a cada momento
→ **elemento corrente**;
- **Acesso aos dados:**
sequencial ou randômico.

Algumas funções de arquivo

A linguagem C não possui comandos de entrada e saída, assim as operações sobre arquivos são realizadas através de funções pré-definidas, algumas delas já vistas:

- `fopen`
- `fclose`
- `fflush`
- `fread`
- `feof`
- `fwrite`

Função *fopen*:

Modos de abertura para arquivos binários

O modo de abertura define as operações que podem ser realizadas sobre um arquivo e o que deve ocorrer se ele não existir previamente.

Modos de abertura para arquivos binários (na especificação do modo aparece a letra b): *rb*, *wb*, *ab*, *r+b*, *w+b*, *a+b*.

Cada modo tem as suas especificidades.

No caso de *rb*, por exemplo, o modo de abertura autoriza exclusivamente leitura e se o arquivo não existir previamente ocorrerá erro.

No caso de *wb*, apenas a escrita está autorizada. Se o arquivo não existir será criado e se já existir será recriado, sendo perdidos os dados anteriores.

No caso de *ab*, se o arquivo existir, os dados são acrescentados ao final.

Cont.

Modos de abertura para arquivos binários

Quadro resumo

Modo de abertura	Arquivo Existe	Arquivo não existe	Permissão de leitura	Permissão de escrita
rb	ok	erro	x	
wb	cria novo, antigo eliminado	cria		x
r+b ou rb+	ok	erro	x	x
w+b ou wb+	cria novo, antigo eliminado	cria	x	x
ab	novos dados acrescentados ao final	cria (como se fosse wb)		x
a+b ou ab+	novos dados acrescentados ao final	cria (como se fosse wb+)	x	x

Função AbreArquivo: abertura de arquivo
com nome fornecido pelo usuário em tempo de execução

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

Se a função *fopen* executar corretamente, ela retornará um ponteiro válido, caso contrário ela retornará NULL

Cont.

Função **fflush**: descarga das áreas intermediárias (*buffers*)

Protótipo: `int fflush (FILE *f)`

Efetiva a gravação no arquivo, em dispositivo de memória auxiliar, dos dados que ainda se encontravam nas áreas intermediárias associadas ao arquivo, sem fechar o fluxo de dados.

Ex.:

```
fflush(arq);
```

Função *fread*: leitura

`fread (&varbuffer,numbytes,quant,FILE *)`

➡ Lê dados do arquivo para a memória principal

Lê dados do arquivo, a partir do elemento corrente, e os armazena em *varbuffer*.

Os parâmetros dessa função são:

- onde os dados lidos devem ser armazenados: *&varbuffer*, que pode ser o endereço de uma variável com o tamanho dos bytes que serão lidos, ou um ponteiro;
- o tamanho em bytes da unidade a ser lida: *numbytes*;
- quantas unidades deverão ser lidas: *quant*;
- o ponteiro para o arquivo sendo lido: *FILE **.

Em uma operação de leitura com *fread* o número total de bytes lidos é: $\text{numbytes} * \text{quant}$.

Retorno da função: o número de unidades lidas.

Que poderá ser menor que o previsto se o final do arquivo for alcançado antes do esperado.

Leitura sequencial de um arquivo binário

```
void listarq(FILE *fpin2)
{
    float x;
    while(fread(&x, sizeof(float), 1, fpin2) != 0)
        printf("\n%6.2f", x);
}
```

Altamente
recomendável!

A função **listarq** lê um arquivo sequencialmente.

O *while* concluirá quando uma tentativa de leitura resultar em zero *bytes* lidos, ou seja, não for bem sucedida por ter sido encontrado o final do arquivo.

Leitura sequencial de um arquivo binário

```
void ListagemAtletas(FILE *arq2, struct atleta buffer)
{
    printf("-----Comeco da listagem-----\n");
    while(!feof(arq2))
        if (fread(&buffer,sizeof(struct atleta),1,arq2) == 1)
            {//le e confirma se o que foi lido eh estrutura,
                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");
}
```

A função **ListagemAtletas** lê um arquivo sequencialmente.

O **while** (!feof(arq2)) concluíra quando for detectado que o arquivo está posicionado além do seu limite em bytes.

Como a situação de fim de arquivo só é detectada quando uma tentativa de leitura é mal sucedida, então é necessário verificar o resultado da operação de leitura a cada nova tentativa.

Execução correta da função ListagemAtletas

Nesta versão, apenas quando a função de leitura executa ok é que dados são apresentados. O arquivo original continha dados de dois atletas e apenas eles são mostrados.

```
C:\backupcida\LinguagemCPagina20082\ArquivosDesenvolvimento20082>
Nome do arquivo (com no maximo 30 caracteres): c:\atleta
-----Comeco da listagem-----
Nome: pedro
Idade: 12
Altura: 1.60

Nome: Rosa
Idade: 24
Altura: 1.90

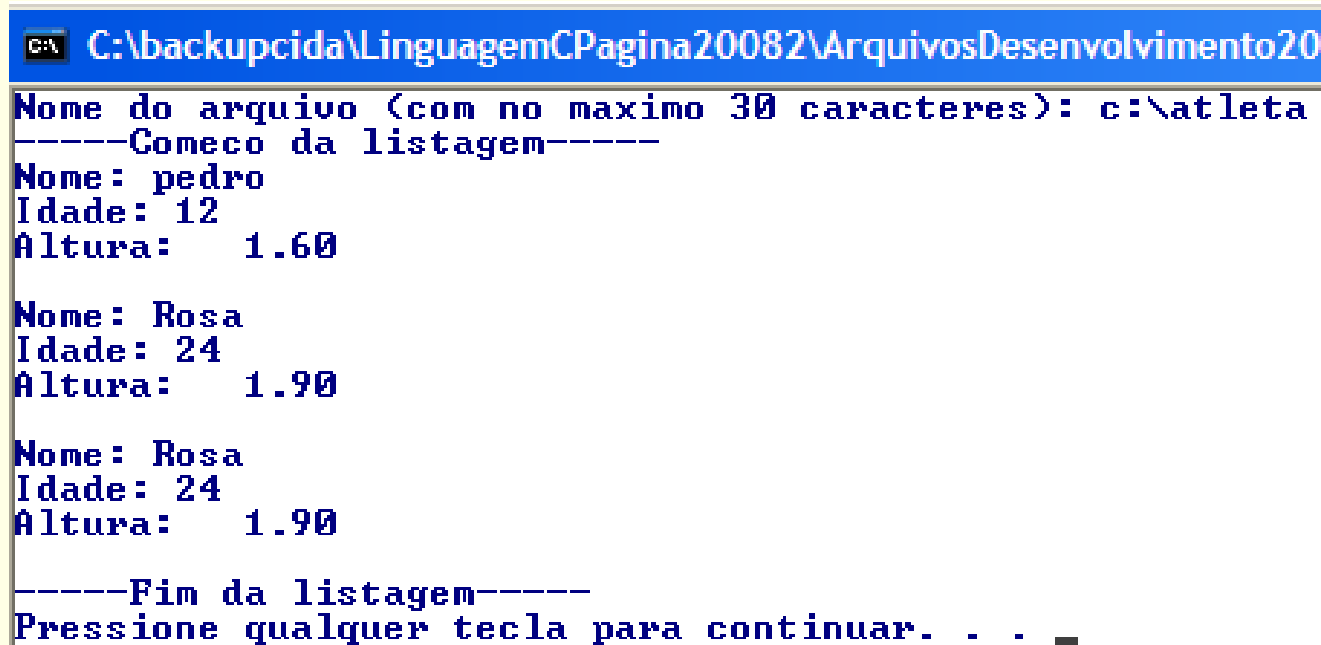
-----Fim da listagem-----
Pressione qualquer tecla para continuar. . . _
```

Execução incorreta da função ListagemAtletas

Na função ListagemAtleta modificou-se o *while*, retirando-se a verificação se a operação de leitura resultava ok:

```
while(!(feof(arq2)))  
{  
    fread(&buffer,sizeof(struct atleta),1,arq2)  
    printf("Nome: %s\n",buffer.nome);  
    printf("Idade: %d\n",buffer.idade);  
    printf("Altura: %6.2f\n\n",buffer.altura);  
}
```

Quando a tentativa de ler ultrapassou o final do arquivo, nada foi lido, mas como as variáveis continham ainda os dados do último atleta processado esses dados foram apresentados novamente.



```
C:\backupcida\LinguagemCPagina20082\ArquivosDesenvolvimento20  
Nome do arquivo (com no maximo 30 caracteres): c:\atleta  
-----Comeco da listagem-----  
Nome: pedro  
Idade: 12  
Altura:  1.60  
  
Nome: Rosa  
Idade: 24  
Altura:  1.90  
  
Nome: Rosa  
Idade: 24  
Altura:  1.90  
  
-----Fim da listagem-----  
Pressione qualquer tecla para continuar. . . _
```


Função `fwrite`: escrita

`fwrite (&varbuffer,numbytes,quant,FILE *)`



Grava blocos de *bytes* em um arquivo, a partir do elemento corrente.

Os parâmetros dessa função são:

- de onde os dados devem ser obtidos: `&varbuffer`, que pode ser o endereço de uma variável ou um ponteiro;
- o tamanho em bytes da unidade a ser gravada: `numbytes`;
- quantas unidades deverão ser gravadas: `quant`;
- o ponteiro para o arquivo sendo gravado: `FILE *`.

Em uma operação de escrita com `fwrite` o número total de bytes gravados é: `numbytes * quant`.

Retorno da função: o número de unidades gravadas.

Podem ocorrer erros durante a gravação.

Abertura e fechamento de arquivos

Recomendações

- As operações de abertura e fechamento de um determinado arquivo deverão ser realizadas de uma de duas formas:
 - ambas na *main*;
 - ambas em uma mesma função.

Em outras palavras, considerando-se um arquivo determinado, não fazer sua abertura na *main* e seu fechamento internamente a uma outra função ou vice-versa.
- **Forte recomendação:** concentrar na *main* as operações de abertura e fechamento de arquivos, para maior clareza quanto à sequência das operações realizadas.

Função **feof**: detecção de fim de arquivo

feof (FILE *)

➔ **Indica se um arquivo chegou ao fim.**
Retorna verdadeiro (não-zero) caso o final tenha sido atingido, falso (zero) caso contrário.

```
... FILE *arq;  
... while(!feof(arq)){ /* Enquanto não for o fim faça... */  
    ...  
}
```

Quando ocorre a situação fim de arquivo?

Sempre que a posição corrente de um arquivo estiver além do número de bytes que ele contém armazenado.

Logo:

se um arquivo vazio for aberto, só vai ser detectado o fim de arquivo após uma tentativa de leitura.

Função **error**: detecção de erro

error(FILE *)

- ➔ **Indica** se a última operação com um arquivo produziu um erro. Cada nova operação com o arquivo modifica a condição de erro.

```
...  
FILE *arq;  
float pilido;  
arq = fopen(...,"rb");  
  
...  
fread(&pilido,...,...,arq);  
if(error(arq))  
    printf("Erro na Leitura!");  
...  
...
```

Operador *Sizeof*

O operador *sizeof* é usado para se saber o tamanho de variáveis ou de tipos.

Ele retorna o tamanho do tipo ou variável em *bytes*.

O *sizeof* é chamado um operador porque ele é substituído pelo tamanho do tipo ou variável *no momento da compilação*.

Ele não é uma função.

O *sizeof* admite duas formas:

sizeof nome_da_variável

sizeof (nome_do_tipo)

Se quisermos então saber o tamanho de um float fazemos *sizeof(float)*.

Se declararmos a variável *f* como float e quisermos saber o seu tamanho faremos *sizeof f*.

Fonte: <http://www.mtm.ufsc.br/~azeredo/cursoC/aulas/cb50.html>

Observações importantes:

Únicas declarações globais autorizadas:

- ponteiros de arquivos (mas que devem ser passados como parâmetros para funções);
- tipo estrutura (só o tipo, variáveis não);
- constantes (mas que devem ser passadas como parâmetros para as funções).

Recomendações:

Determinação de tamanho de variável: usar *sizeof*.

Nomes de funções: usar maiúsculas e minúsculas.

Se usar *fgets* para ler nomes de arquivos: se o último caractere armazenado (posição $\text{strlen}(\text{nome}) - 1$) for `\n`, alterar para `\0` (conforme exemplos).

Os exemplos a seguir, **todos,**
processam **arquivos binários com**
acesso sequencial.

O acesso randômico a arquivos
binários será visto proximamente.

Exemplo 1: Criação e escrita de um arquivo com valores reais (float)

```
/*  
  Cria um arquivo de reais (float) positivos de forma sequencial.  
  Apos, lista o conteúdo do arquivo também sequencialmente.  
*/  
#include <stdio.h>  
#include <string.h>  
#include <stdlib.h>  
FILE *fpin; //ponteiro para o arquivo  
void criarq(FILE *);  
void listarq(FILE *);  
int main ( )  
{  
  char filename[20];  
  float x;  
  system("color f1");  
  if (!(fpin = fopen("c:\\numfloat", "wb")))//verifica ocorrencia de erro na  
                                             //abertura do arquivo  
      printf("\nO arquivo nao pode ser aberto!");  
  else  
  {  
    printf("\nCriacao do arquivo sequencial c:\\numfloat");  
    criarq(fpin);  
    fclose(fpin);  
  }  
}
```

Modo de abertura
exclusivamente
para escrita

Duas barras \\ para que
uma seja apresentada.

cont.

Exemplo 1: Criação e escrita de um arquivo com valores reais (float) (cont.)

Modo de abertura exclusivamente para leitura.

```
system("pause");
if (!(fpin=fopen("c:\\numfloat", "rb"))){
    printf("Nao foi possivel abrir o arquivo. Trabalho cancelado.\n");
    system("pause");
}
else{
    listarq(fpin);
    fclose(fpin);
    printf("\n\n");
    system("pause");
    return 0;
}
```

cont.

Exemplo 1: Criação e escrita de um arquivo com valores reais (float) (cont.)

```
void criarq(FILE *fpin2)
{
    float x;
    printf("\nEntre com valores reais positivos");
    printf("\nNum. real (negativo para parar): ");
    scanf("%f",&x);
    while(x >=0)
    {
        fwrite(&x, sizeof(float), 1, fpin2);
        printf("\nNum. real(negativo para parar): ");
        scanf("%f",&x);
    }
}

void listarq(FILE *fpin2)
{
    float x;
    while(fread(&x, sizeof(float), 1, fpin2) != 0)
        printf("\n%6.2f ", x);
}
```

Enquanto a função de leitura estiver recuperando dados do arquivo, o *while* segue.

Execução do exemplo 1:

C:\backupcida\LinguagemCPagina20082\ArquivosDesenvolv

Criacao do arquivo sequencial c:\numfloat

Entre com valores reais positivos

Num. real (negativo para parar): 1.2

Num. real(negativo para parar): 3.4

Num. real(negativo para parar): 5.6

Num. real(negativo para parar): 7.8

Num. real(negativo para parar): -4

Pressione qualquer tecla para continuar. . .

1.20

3.40

5.60

7.80

Pressione qualquer tecla para continuar. . .

Exemplo 2: Criação de um arquivo com dados de atletas armazenados em uma estrutura.

```
/*  
Cria de forma sequencial um arquivo com dados de uma  
estrutura com informacoes de atletas.  
*/  
#include <stdio.h>  
#include <stdlib.h>  
#include <string.h>  
#define MAXNOME 31  
struct atleta  
{  
    char nome[MAXNOME];  
    int idade;  
    float altura;  
};  
FILE *arq;  
void RotinaInsere(FILE *, struct atleta, int);  
FILE * AbreArquivo(int max);
```

cont.

Exemplo 2: Criação de um arquivo com dados de atletas armazenados em uma estrutura (cont.)

```
int main( )
{
    char nome[15];
    struct atleta buffermain;
    int op;
    system("color f1");
    if(!(arq = AbreArquivo(MAXNOME)))
    {
        printf("Erro na abertura");
        system("pause");
    }
    else
    {
        do
        {
            RotinaInsere(arq, buffermain, MAXNOME);
            printf("\n1-InserirNovo, 2-Encerrar\n");
            scanf("%d", &op);
        }
        while(op != 2);
        fclose(arq);
        system("pause");
        return 0;
    }
}
```

cont.

Exemplo 2: Criação de um arquivo com dados de atletas armazenados em uma estrutura (cont.)

```
void RotinaInsere(FILE *arq2, atleta buffer, int max)
{
    fflush(stdin);
    printf("\nNome (com no maximo %d caracteres): ", max - 1);
    fgets (buffer.nome, sizeof(buffer.nome), stdin);
    fflush(stdin);
    printf("\nIdade: ");
    scanf("%d",&buffer.idade);
    printf("\nAltura: ");
    scanf("%f",&buffer.altura);
    fwrite(&buffer,sizeof(struct atleta),1,arq2);
}

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

Execução do exemplo 2:

```
C:\backupcida\LinguagemCPagina20082\ArquivosDesenvolvimento20082>
Nome do arquivo (com no maximo 30 caracteres): c:\atleta
Nome (com no maximo 30 caracteres): Maria da Silva
Idade: 23
Altura: 1.7
1-InserirNovo, 2-Encerrar
1
Nome (com no maximo 30 caracteres): Pedro Cardozo
Idade: 22
Altura: 1.92
1-InserirNovo, 2-Encerrar
2
Pressione qualquer tecla para continuar. . . _
```

Exemplo 3: Listagem de um arquivo com dados de atletas armazenados em uma estrutura.

```
/*  
  Lista sequencialmente um arquivo com dados de uma estrutura  
  com informacoes de atletas.  
*/  
#include <stdio.h>  
#include <stdlib.h>  
#include <string.h>  
#define MAXNOME 31  
struct atleta  
{  
    char nome[MAXNOME];  
    int idade;  
    float altura;  
};  
  
FILE *arq;  
void ListagemAtletas(FILE *, struct atleta);  
FILE * AbreArquivo(int max);
```

cont.

Exemplo 3: Listagem de um arquivo com dados de atletas armazenados em uma estrutura.

```
int main( )
{
    char nome[15];
    struct atleta buffermain;
    if(!(arq = AbreArquivo(MAXNOME)))
    {
        printf("Erro na abertura\n");
        system("pause");
    }
    else
    {
        ListagemAtletas(arq, buffermain);
        fclose(arq);
        system("pause");
        return 0;
    }
}
```

Exemplo 3: Listagem de um arquivo com dados de atletas armazenados em uma estrutura (cont.).

```
void ListagemAtletas(FILE *arq2, struct atleta buffer)
{
    printf("-----Comeco da listagem      \n");
    while(!feof(arq2))
        if (fread(&buffer, sizeof(struct atleta), 1, arq2) == 1)
            {//le e confirma se o que foi lido eh estrutura, entao imprime
                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");
}

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

O final de arquivo só é detectado quando uma leitura é tentada e não encontra mais dados. Por isso a leitura é verificada, para ver se resultou ok.

Execução do exemplo 3:

```
C:\backupcida\LinguagemCPagina20082\ArquivosDesenvolvimento2008
Nome do arquivo (com no maximo 30 caracteres): c:\atleta
-----Comeco da listagem-----
Nome: Maria da Silva
Idade: 23
Altura: 1.70

Nome: Pedro Cardozo
Idade: 22
Altura: 1.92

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

Exemplo 4: Apresentação da altura de um atleta cujo nome é fornecido pelo usuário.

```
//Apresenta a altura de um atleta cujo nome eh fornecido pelo usuario.
#include <stdio.h>
#include <stdlib.h>
#include <conio.h>
#include <string.h>
#define MAXNOME 31
FILE *arq;
struct atleta
{
    char nome[MAXNOME];
    int idade;
    float altura;
};
int main( )
{
    struct atleta buffer;
    char procurado[MAXNOME], nomearq[MAXNOME];
    int encontrado, contador = 0;
    system("color f1");
    printf("Nome do arquivo (com no maximo %d caracteres): ", MAXNOME - 1);
    fgets(nomearq, sizeof(nomearq), stdin);
    if (nomearq[strlen(nomearq) - 1] == '\n')
        nomearq[strlen(nomearq) - 1] = '\0';
    fflush(stdin);
    (...)
```

cont.

Exemplo 4: Apresentação da altura de um atleta cujo nome é fornecido pelo usuário (cont).

```
(...)  
if(! (arq = fopen(nomearq,"rb")))  
{  
    printf("Erro abertura");  
    system("pause");  
}  
else  
{  
    printf("Nome do procurado (maximo %d caracteres): ", MAXNOME - 1);  
    fgets(procurado, sizeof(procurado), stdin);  
    fflush(stdin);  
    while(!feof(arq))  
    {  
        if (fread(&buffer,sizeof(struct atleta),1,arq) == 1)  
            if (!(strcmp(buffer.nome,procurado)))  
            {  
                printf("\nAltura do procurado: %.2f",buffer.altura);  
                encontrado = 1; /*verdade, alguém foi encontrado*/  
                contador++;  
            }  
        }  
    if (!encontrado)  
        printf("\nNenhum atleta encontrado");  
    else  
        printf("\nOcorrencias encontradas: %d",contador);  
    fclose(arq);  
    printf("\n\n");  
    system("pause");  
    return 0;  
}  
}
```

Execução do exemplo 4:

```
C:\ C:\backupcida\LinguagemCPagina20082\ArquivosDesenvolvimento20082\
Nome do arquivo (com no maximo 30 caracteres): C:\ATLETA
Nome do procurado (maximo 30 caracteres): Maria da Silva

Altura do procurado: 1.70
Ocorrencias encontradas: 1

Pressione qualquer tecla para continuar. . . _
```

Exemplo 5: Inserção de mais dados de atletas no fim do arquivo.

//Acrescenta dados ao final de um arquivo previamente existente

```
#include <stdio.h>
#include <stdlib.h>
#include <conio.h>
#include <string.h>
#define MAXNOME 31
FILE *arq;
struct atleta
{
    char nome[MAXNOME];
    int idade;
    float altura;
};
int main( )
{
    struct atleta buffer;
    char nome[15];
    int op;
    system("color f1");
    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);
```

cont.

Exemplo 5: Inserção de mais dados de atletas no fim do arquivo (cont.).

```
if(!(arq = fopen(nome, "ab"))){
    printf("Erro na abertura para acrescentar");
    system("pause");
}
else {
    do {
        printf("\nNome: ");
        fflush(stdin);
        fgets(buffer.nome, sizeof(buffer.nome), stdin);
        fflush(stdin);
        printf("\nIdade: ");
        scanf("%d", &buffer.idade);
        printf("\nAltura: ");
        scanf("%f", &buffer.altura);
        fwrite(&buffer, sizeof(struct atleta), 1, arq);
        printf("\n1-Inserir Novo, 2-Encerrar\n");
        scanf("%d", &op);
    }
    while(op != 2);
    fclose(arq);
    system("pause");
    return 0;
}
}
```

Modo de abertura para acrescentar dados ao final do arquivo.

Arquivo antes da execução do exemplo 5

```
C:\backupcida\LinguagemCPagina20082\ArquivosDesenvolvimento20082
Nome do arquivo (com no maximo 30 caracteres): c:\atleta
-----Comeco da listagem-----
Nome: Oto Schmidt
Idade: 24
Altura: 1.78
Nome: Vera Souza
Idade: 22
Altura: 1.90
Nome: Pedro Antunes
Idade: 23
Altura: 1.89
Nome: Valkyria
Idade: 21
Altura: 2.10
-----Fim da listagem-----
Pressione qualquer tecla para continuar. . .
```

Execução do exemplo 5

```
C:\backupcida\LinguagemCPagina20082\ArquivosDesenvolvimento20082
Nome do arquivo (com no maximo 30 caracteres): c:\atleta
Nome: Rosa Maria Lima
Idade: 20
Altura: 1.78
1-InserirNovo, 2-Encerrar
1
Nome: Ana Lucia Viera
Idade: 20
Altura: 1.90
1-InserirNovo, 2-Encerrar
2
Pressione qualquer tecla para continuar. . .
```

Arquivo após execução do exemplo 5

```
C:\backupcida\LinguagemCPagina20082\ArquivosDesenvolvimento20082
Nome do arquivo (com no maximo 30 caracteres): c:\atleta
-----Comeco da listagem-----
Nome: Oto Schmidt
Idade: 24
Altura: 1.78
Nome: Vera Souza
Idade: 22
Altura: 1.90
Nome: Pedro Antunes
Idade: 23
Altura: 1.89
Nome: Valkyria
Idade: 21
Altura: 2.10
Nome: Rosa Maria Lima
Idade: 20
Altura: 1.78
Nome: Ana Lucia Viera
Idade: 20
Altura: 1.90
-----Fim da listagem-----
Pressione qualquer tecla para continuar. . .
```

Os exemplos a seguir, 6, 6.1 e 6.2, estão relacionados.

Todos trabalham com arquivos de inteiros.

O 6 cria um arquivo de inteiros, o 6.1 apresenta um arquivo de inteiros e o 6.2 lê um arquivo de inteiros e cria um outro, apenas com os valores pares do arquivo lido.

// Cria um arquivo de inteiros de forma sequencial.

#include <stdio.h>

#include <string.h>

#include <stdlib.h>

FILE *fpin;

int main ()

{

int x, segue = 1;

system("color f1");

if ((fpin=fopen("c:\\numint","wb"))==NULL)

printf("\nO arquivo nao pode ser aberto!");

else

{

printf("\nCriacao do arquivo sequencial c:\\numint");

printf("\nEntre com valores inteiros");

while(segue)

{

printf("\nNum. inteiro: ");

scanf("%d",&x);

fwrite(&x, sizeof(int), 1, fpin);

printf("\n1 - Segue; 0 - Para\n");

scanf("%d",&segue);

}

fclose(fpin);

}

system("pause");

return 0;

}

Exemplo 6: Cria um arquivo com inteiros.

Execução do exemplo 6:

```
C:\ C:\backupcida\LinguagemCPagina20082\ArquivosDesenv  
  
Criacao do arquivo sequencial c:\numint  
Entre com valores inteiros  
Num. inteiro: 1  
  
1 - Segue; 0 - Para  
1  
  
Num. inteiro: 2  
  
1 - Segue; 0 - Para  
1  
  
Num. inteiro: 3  
  
1 - Segue; 0 - Para  
1  
  
Num. inteiro: 41  
  
1 - Segue; 0 - Para  
0  
Pressione qualquer tecla para continuar. . . _
```

// Lista um arquivo de inteiros.

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

```
#include <string.h>
```

```
#include <stdlib.h>
```

```
#define MAXNOME 21
```

```
FILE *fpin;
```

```
int main ( )
```

```
{
```

```
    char filename[MAXNOME];
```

```
    int x;
```

```
    system("color f1");
```

```
    printf("Nome do arquivo (com no maximo %d caracteres): ", MAXNOME - 1);
```

```
    fgets(filename, sizeof(filename), stdin);
```

```
    if (filename[strlen(filename) - 1] == '\n')
```

```
        filename[strlen(filename) - 1] = '\0';
```

```
    fflush(stdin);
```

```
    if ((fpin=fopen(filename,"rb"))==NULL)
```

```
    {
```

```
        printf("Nao foi possivel abrir o arquivo %s. Trabalho cancelado.\n",filename);
```

```
        system("pause");
```

```
    }
```

```
    else
```

```
    {
```

```
        while(fread(&x, sizeof(int),1,fpin) !=0)
```

```
            printf("\n%4d ",x);
```

```
            printf("\n\n");
```

```
            system("pause");
```

```
            return 0;
```

```
        }
```

```
}
```

Exemplo 6.1: Lista um arquivo com inteiros.

Execução do exemplo 6.1:

```
C:\backupcida\LinguagemCPagina20082\ArquivosDesenvolvimento2008
Nome do arquivo (com no maximo 20 caracteres): c:\numint
1
2
3
41
Pressione qualquer tecla para continuar. . .
```

Exemplo 6.2: Cria um arquivo com inteiros a partir de outro já existente.

```
//Cria um arquivo de inteiros apenas com valores pares, a
//partir de outro já existente.
#include <stdio.h>
#include <string.h>
#include <stdlib.h>
FILE *fpin;
FILE *fppar;
int main ( )
{
    int x , contgrav = 0,  contlidos = 0;
    system("color f1");
    printf("\nArquivo lido: c:\\numint\n");
    printf("\nArquivo gravado: c:\\numintpar\n");
    if ((fpin=fopen("c:\\numint","rb"))==NULL)
        printf("\nO arquivo de entrada nao pode ser aberto!\n");
    else
        (...)
```

cont.

Exemplo 6.2: Cria um arquivo com inteiros a partir de outro já existente (cont.)

```
(...)  
{  
    if ((fppar = fopen("c:\\numintpar", "wb")) == NULL)  
        printf("\nO arquivo de saida nao pode ser aberto!\n");  
    else  
        while(fread(&x, sizeof(int), 1, fpin) != 0)  
        {  
            contlidos++;  
            if (!(x % 2))  
            {  
                fwrite(&x, sizeof(int), 1, fppar);  
                contgrav++;  
            }  
        }  
        printf("\nLidos: %d\n", contlidos);  
        printf("\nGravados: %d\n", contgrav);  
        fclose(fpin);  
        fclose(fppar);  
        system("pause");  
        return 0;  
    }  
    system("pause");  
    return 0;  
}
```


Execução do exemplo 6.2:

```
C:\backupcida\LinguagemCPagina20082\ArquivosDeser
Arquivo lido: c:\numint
Arquivo gravado: c:\numintpar
Lidos: 4
Gravados: 1
Pressione qualquer tecla para continuar. . .
```

Listagem de numintpar pelo programa que lista arquivo de inteiros (exemplo 6.1):

```
C:\ C:\backupcida\LinguagemCPagina20082\ArquivosDesenvolvimento20082\
Nome do arquivo (com no maximo 20 caracteres): c:\numintpar
2
Pressione qualquer tecla para continuar. . . _
```

Exemplo 7: Lê e escreve nomes de grupos de pessoas

```
/*  
Le e escreve nomes de grupos de pessoas  
*/  
#include <stdio.h>  
#include <stdlib.h>  
#define MAXNOME 31  
#define MAXPART 4  
struct gr  
{  
    int num_part_gr;  
    char nome_part[MAXNOME][MAXPART];  
};  
FILE * grupos;  
void LeGrupo(struct gr *, int , int);  
void ApresentaGrupo (struct gr *);  
int main( )  
{  
    int segue = 1, contgrv = 0, contlidos = 0;  
    struct gr grupo;  
    system("color f1");  
    if (!(grupos = fopen ("c:\\grupos", "wb")))  
        printf("Nao abriu arquivo para escrita!");  
    else  
    {
```

Vetor de strings



cont.

Exemplo 7: Lê e escreve nomes de grupos de pessoas (cont.)

```
else
{
    while (segue)
    {
        LeGrupo(&grupo, MAXNOME, MAXPART);
        fwrite (&grupo, sizeof(gr), 1, grupos);
        contgrv++;
        printf("\n1- Segue , 0 - Para\n");
        scanf("%d", &segue);
    }
    printf("\nGravados = %d\n", contgrv);
    fclose (grupos);
    if (!(grupos = fopen ("c:\\grupos", "rb")))
        printf("Nao abriu arquivo para leitura!");
    else
    {
        while (!feof(grupos))
            if (fread(&grupo, sizeof(grupo), 1, grupos) == 1)
            {
                ApresentaGrupo(&grupo);
                contlidos++;
            }
        fclose(grupos);
        printf("\nLidos = %d\n", contlidos);
        system("pause");
        return 0;
    }
}
```

cont.

Exemplo 7: Lê e escreve nomes de grupos de pessoas (cont.)

```
void LeGrupo(struct gr * grup, int maxnom , int maxpart)
{
    int i;
    fflush(stdin);
    do
    {
        printf("\nNumero de participantes do grupo (maximo = %d): ", maxpart);
        scanf("%d", &(*grup).num_part_gr);
        if ((*grup).num_part_gr < 1 || (*grup).num_part_gr > maxpart)
            printf
                ("\nNumero participantes deve ser > 0 e no maximo %d", maxpart);
    }
    while ((*grup).num_part_gr < 1 || (*grup).num_part_gr > maxpart);
    for (i = 0; i < (*grup).num_part_gr; i++)
    {
        printf("\nNome do participante %d: ", i + 1);
        fflush(stdin);
        fgets((*grup).nome_part[i], sizeof((*grup).nome_part), stdin) ;
        fflush(stdin);
    }
}
```

cont.

Exemplo 7: Lê e escreve nomes de grupos de pessoas (cont.)

```
void ApresentaGrupo (struct gr *grup)
{
    int i;
    printf
        ("\n\nNumero de participantes da grupo: %d ", (*grup).num_part_gr);
    for (i = 0; i < (*grup).num_part_gr; i++)
        printf("\nNome do participante %d: %s", i + 1, (*grup).nome_part[i]);
}
```

Execução do exemplo 7:

```
C:\F:\ARQUIVOS20091\ITEM18\Exemplo07ApresentacaoArquivos02.e
Numero de participantes do grupo (maximo = 4): 4
Nome do participante 1: Ana
Nome do participante 2: Pedro
Nome do participante 3: Rosa
Nome do participante 4: Oto
1- Segue , 0 - Para
1
Numero de participantes do grupo (maximo = 4): 1
Nome do participante 1: Maria
1- Segue , 0 - Para
1
Numero de participantes do grupo (maximo = 4): 3
Nome do participante 1: Luis
Nome do participante 2: Vera
Nome do participante 3: Carlos
1- Segue , 0 - Para
0
Gravados = 3

Numero de participantes da grupo: 4
Nome do participante 1: Ana
Nome do participante 2: Pedro
Nome do participante 3: Rosa
Nome do participante 4: Oto

Numero de participantes da grupo: 1
Nome do participante 1: Maria

Numero de participantes da grupo: 3
Nome do participante 1: Luis
Nome do participante 2: Vera
Nome do participante 3: Carlos
Lidos = 3
Pressione qualquer tecla para continuar. . .
```