ARQUIVOS

Prof. Muriel Mazzetto

Algoritmos 2

Entrada/Saída com Arquivos

- Em C, as operações de entrada e saída utilizam funções da biblioteca padrão *stdio*.
- **Dispositivos de E/S:** Sdtin, Stdout, Arquivos.

Entrada/Saída com Arquivos

- Em C, as operações de entrada e saída utilizam funções da biblioteca padrão *stdio*.
- **Dispositivos de E/S:** Sdtin, Stdout, Arquivos.
- **Tipos de dados:** simples (int, double, char...), vetores e registros (struct).

Entrada/Saída com Arquivos

- Em C, as operações de entrada e saída utilizam funções da biblioteca padrão *stdio*.
- **Dispositivos de E/S:** Sdtin, Stdout, Arquivos.
- **Tipos de dados:** simples (int, double, char...), vetores e registros (struct).
- **Tipos de Arquivos:**
  - Texto: converte os bytes em dados no padrão ASCI ao arquivá-los, tornando legível para o usuário (.txt);
  - Binário: arquiva os bytes referentes aos dos dados, não legível pelo usuário (.dat, .bin).

Funções

Nome	Função
fopen()	Abre um arquivo
fclose()	Fecha um arquivo
fputc() / fputs()	Escreve um/vários caractere(s) (putc/puts)
fprintf()	printf() em arquivos
fwrite()	Escreve um bloco de informações no arquivo binário (structs e vetores)
fgetc() / fgets()	Lê um/vários caractere(s) (getc/gets)
fscanf()	scanf() em arquivos
fread()	Lê um bloco de informações do arquivo binário (structs e vetores)
feof()	Informa se atingiu o final do arquivo
ferror()	Informa se ocorreu erro na gravação/leitura

Funções

Nome	Função
rewind()	Realoca o indicador de posição no início do arquivo (voltar ao início)
remove()	Apaga um arquivo
fseek()	Reposicionar o indicador de posição em um local específico do arquivo

Trabalhando com Arquivos

- Criar ponteiro para receber arquivo;
- Abrir o arquivo;
- Editar o arquivo (Ler/Escrever);
- Fechar o arquivo.

Ponteiro de arquivo

- O ponteiro de arquivo recebe suas características, como nome, status e posição atual.
- É uma variável do tipo FILE. O ponteiro é obtido pelo comando:
  - FILE *fp; //fp é ponteiro de arquivo

Abrir arquivo

- Após ter um ponteiro para receber os dados do arquivo, é necessário abrir ou criar um.
- A função fopen() acessa um diretório e retorna um endereço para o arquivo, de acordo com o modo selecionado.

Abrir arquivo

- Após ter um ponteiro para receber os dados do arquivo, é necessário abrir ou criar um.
- A função fopen() acessa um diretório e retorna um endereço para o arquivo, de acordo com o modo selecionado.
- O escopo da função é:
  - **FILE* fopen(const char *_dir_nome_, const char* _modo_);**
  - *dir_nome* é um ponteiro para uma string, que define o nome do arquivo ou diretório.
  - *modo* define quais operações serão realizadas sobre o arquivo.

Abrir arquivo

- Após ter um ponteiro para receber os dados do arq
- A
- O
  - *modo* define quais operações serão realizadas sobre o arquivo.

SE FOR INFORMADO APENAS O NOME DO ARQUIVO, ELE DEVE ESTAR NA MESMA PASTA QUE O EXECUTÁVEL.

SE ESTIVEREM EM PASTAS DIFERENTES, PODE SER INFORMADO O CAMINHO DO DIRETÓRIO PARA ACESSAR O ARQUIVO.

Abrir arquivo

Modo	Significado
r	Abre um arquivo de texto para leitura (deve existir)
w	Cria um arquivo de texto para escrita (sobrescreve)
a	Concatena a um arquivo de texto (escreve no fim)
rb	Abre um arquivo binário para leitura
wb	Cria um arquivo de binário para escrita
ab	Concatena a um arquivo binário
r+	Abre um arquivo de texto para leitura/escrita
w+	Cria um arquivo de texto para leitura/escrita
a+	Concatena/Cria arquivo de texto para leitura/escrita
rb+	Abre um arquivo binário para leitura/escrita
wb+	Cria um arquivo binário para leitura/escrita
ab+	Concatena/Cria arquivo binário para leitura/escrita

Abrir arquivo

- Trecho de código para abrir um arquivo

```
//criar ponteiro de arquivo
FILE *arquivo_txt;
//abrir arquivo de texto para escrever
arquivo_txt = fopen("dados.txt","w");
//verificar se foi alocado e aberto
if(arquivo_txt==NULL)
{
    printf("NAO FOI POSSIVEL ABRIR O ARQUIVO!\n");
    return 0;
}
```

Fechar arquivo

- Após realizar as edições no arquivo, é necessário fechá-lo, para não extrapolar a quantidade limite de arquivos abertos.
- Garante que as informações foram salvas.

Fechar arquivo

- Após realizar as edições no arquivo, é necessário fechá-lo, para não extrapolar a quantidade limite de arquivos abertos.
- Garante que as informações foram salvas.
- A função fclose() recebe o ponteiro do arquivo e retorna 0 se a operação foi bem sucedida.
- O escopo é:
  - **int fclose(FILE *fp);**

Fechar arquivo

- Trecho de código para fechar um arquivo

```
//fechar arquivo
fclose(arquivo_txt);
```

- OBS: é comum verificar se o retorno da função é diferente de 0, isso indica erro ao tentar fechar ou salvar o arquivo.

Editando o arquivo: Leitura

- Para ler os dados de um arquivo, as funções mais comuns são:
  - fscanf(): funciona igual um scanf(), porém recebe um parâmetro a mais, que é o ponteiro para o arquivo.
  - fread(): utilizada em arquivos binários, para ler um conjunto completo de dados (vetores e estruturas).

Editando o arquivo: Leitura

int fscanf(FILE *fp, const char *string, ...);

- Exemplo:

```
for(i = 0; i < tam; i++)
{
    fscanf(arquivo_txt," %s",vetor[i].nome);
    fscanf(arquivo_txt,"%d",&vetor[i].idade);
}
```

Editando o arquivo: Leitura

int fscanf(FILE *fp, const char *string, ...);

- Exemplo:

```
for(i = 0; i < tam; i++)
{
    fscanf(arquivo_txt," %s",vetor[i].nome);
    fscanf(arquivo_txt,"%d",&vetor[i].idade);
}
```

IGUAL UM SCANF() PARA LER OS VALORES.

Editando o arquivo: Leitura

int fscanf(FILE *fp, const char *string, ...);

- Exemplo:

```
for(i = 0; i < tam; i++)
{
    fscanf(arquivo_txt," %s",vetor[i].nome);
    fscanf(arquivo_txt,"%d",&vetor[i].idade);
}
```

PONTEIRO DO ARQUIVO DE ENTRADA, QUE SERÁ LIDO PELA FUNÇÃO.

Editando o arquivo: Leitura

size_t fread(void* ptr, size_t bytes, size_t qtd, FILE *fp);

- Exemplo:

```
//ler o arquivo para o vetor
fread(vetor,sizeof(dados),qtd,arquivo_bin);
```

Editando o arquivo: Leitura

size_t fread(void* ptr, size_t bytes, size_t qtd, FILE *fp);

- Exemplo:

```
//ler o arquivo para o vetor
fread(vetor,sizeof(dados),qtd,arquivo_bin);
```

- Permite ler um vetor completo com uma linha de comando.

Editando o arquivo: Leitura

size_t fread(void* ptr, size_t bytes, size_t qtd, FILE *fp);

- Exemplo:

```
//ler o arquivo para o vetor
fread(vetor,sizeof(dados),qtd,arquivo_bin);
```

PONTEIRO DO VETOR QUE IRÁ ARMAZENAR OS DADOS LIDOS (nome do vetor).

Editando o arquivo: Leitura

size_t fread(void* ptr, size_t bytes, size_t qtd, FILE *fp);

- Exemplo:

```
//ler o arquivo para o vetor
fread(vetor,sizeof(dados),qtd,arquivo_bin);
```

TAMANHO DO TIPO DE DADO QUE SERÁ LIDO DO ARQUIVO.

Editando o arquivo: Leitura

size_t fread(void* ptr, size_t bytes, size_t qtd, FILE *fp);

- Exemplo:

```
//ler o arquivo para o vetor
fread(vetor,sizeof(dados),qtd,arquivo_bin);
```

QUANTIDADE DE VALORES QUE SERÃO LIDOS, NÃO PODEM EXTRAPOLAR O TAMANHO DO VETOR.

Editando o arquivo: Leitura

size_t fread(void* ptr, size_t bytes, size_t qtd, FILE *fp);

- Exemplo:

```
//ler o arquivo para o vetor
fread(vetor,sizeof(dados),qtd,arquivo_bin);
```

PONTEIRO DO ARQUIVO DE ENTRADA, QUE SERÁ LIDO PELA FUNÇÃO.

Editando o arquivo: Escrita

- Para escrever os dados em um arquivo, as funções mais comuns são:
  - fprintf(): funciona tal qual um printf(), porém recebe um parâmetro a mais, que é o ponteiro para o arquivo.
  - fwrite(): utilizada em arquivos binários, para escrever um conjunto completo de dados (vetores e estruturas).

Editando o arquivo: Escrita

int fprintf(FILE *fp, const char *string, ...);

- Exemplo:

```
//escrever um dado por vez
for(i = 0; i < tam; i++)
{
    fprintf(arquivo_txt,"%s %d\n", vetor[i].nome, vetor[i].idade);
}
```

Editando o arquivo: Escrita

int fprintf(FILE *fp, const char *string, ...);

- Exemplo:

```
//escrever um dado por vez
for(i = 0; i < tam; i++)
{
    fprintf(arquivo_txt,"%s %d\n", vetor[i].nome, vetor[i].idade);
}
```

IGUAL UM PRINTF() PARA ESCREVER OS VALORES NA TELA.

Editando o arquivo: Escrita

int fprintf(FILE *fp, const char *string, ...);

- Exemplo:

```
//escrever um dado por vez
for(i = 0; i < tam; i++)
{
    fprintf(arquivo_txt,"%s %d\n", vetor[i].nome, vetor[i].idade);
}
```

PONTEIRO DO ARQUIVO QUE SERÁ ESCRITO PELA FUNÇÃO.

Editando o arquivo: Escrita

size_t fwrite(void* ptr, size_t bytes, size_t qtd, FILE *fp);

- Exemplo:

```
//escrever o tamanho na primeira linha
fwrite(&tam,sizeof(int),1,arquivo_bin);

//escrever no arquivo binário
fwrite(vetor,sizeof(dados),tam,arquivo_bin);
```

Editando o arquivo: Escrita

size_t fwrite(void* ptr, size_t bytes, size_t qtd, FILE *fp);

- Exemplo:

```
//escrever o tamanho na primeira linha
fwrite(&tam,sizeof(int),1,arquivo_bin);

//escrever no arquivo binário
fwrite(vetor,sizeof(dados),tam,arquivo_bin);
```

- Permite escrever um vetor completo com uma linha de comando.

Editando o arquivo: Escrita

size_t fwrite(void* ptr, size_t bytes, size_t qtd, FILE *fp);

- Exemplo:

```
//escrever o tamanho na primeira linha
fwrite(&tam,sizeof(int),1,arquivo_bin);

//escrever no arquivo binário
fwrite(vetor,sizeof(dados),tam,arquivo_bin);
```

PONTEIRO DO DADO QUE SERÁ ESCRITO (NOME DO VETOR OU ENDEREÇO DA VARIÁVEL).

Editando o arquivo: Escrita

size_t fwrite(void* ptr, size_t bytes, size_t qtd, FILE *fp);

- Exemplo:

```
//escrever o tamanho na primeira linha
fwrite(&tam,sizeof(int),1,arquivo_bin);

//escrever no arquivo binário
fwrite(vetor,sizeof(dados),tam,arquivo_bin);
```

TAMANHO DO TIPO DE DADO QUE SERÁ GRAVADO NO ARQUIVO.

Editando o arquivo: Escrita

size_t fwrite(void* ptr, size_t bytes, size_t qtd, FILE *fp);

- Exemplo:

```
//escrever o tamanho na primeira linha
fwrite(&tam,sizeof(int),1,arquivo_bin);

//escrever no arquivo binário
fwrite(vetor,sizeof(dados),tam,arquivo_bin);
```

QUANTIDADE DE VALORES QUE SERÃO ESCRITOS NO ARQUIVO. PODE SER TODO O VETOR.

Editando o arquivo: Escrita

size_t fwrite(void* ptr, size_t bytes, size_t qtd, FILE *fp);

- Exemplo:

```
//escrever o tamanho na primeira linha
fwrite(&tam,sizeof(int),1,arquivo_bin);

//escrever no arquivo binário
fwrite(vetor,sizeof(dados),tam,arquivo_bin);
```

PONTEIRO DO ARQUIVO QUE SERÁ ESCRITO PELA FUNÇÃO.

Editando o arquivo: Movimentação

- Para mover a localização do apontador de posição:
  - int fseek(FILE *_fp_, long _numbytes_, int _origem_);
  - fp é o ponteiro do arquivo (será alterado);
  - _numbytes_ é a quantidade de bytes que se deslocará o ponteiro;
  - _origem_ é a partir de onde se deslocará os _numbytes_.

ORIGEM	MACRO
Início do arquivo	SEEK_SET
Posição atual	SEEK_CUR
Final do arquivo	SEEK_END

Editando o arquivo

- Para verificar o fim do arquivo:
  - int feof(FILE *fp);
  - Recebe o ponteiro do arquivo e retorna 1 se não existir mais conteúdo para ser lido.

Editando o arquivo

- Para verificar o fim do arquivo:
  - int feof(FILE *fp);
  - Recebe o ponteiro do arquivo e retorna 1 se não existir mais conteúdo para ser lido.

```
lidos = 0;
while(!feof(arquivo_txt) && lidos < tam)
{
    fscanf(arquivo_txt," %s",vetor[lidos].nome);
    fscanf(arquivo_txt,"%d",&vetor[lidos].idade);
    lidos++;
}
```