ARQUIVOS - PROFESSOR: *LEONARDO VIANNA*

1) Funções para manipulação de arquivos (*stdio.h*):

- fopen — abre um arquivo
- fclose — fecha um arquivo
- putc — escreve um caractere em um arquivo
- fputc — o mesmo que putc()
- getc — lê um caractere em um arquivo
- fgetc — o mesmo que getc()
- fseek — posiciona o arquivo em um byte específico
- fprintf — "printf() em arquivo"
- fscanf — "scanf() em arquivo"
- feof — diz se o fim do arquivo foi atingido
- ferror — diz se ocorreu erro na manipulação do arquivo
- rewind — coloca o "cursor" no início do arquivo
- remove — apaga um arquivo
- rename — renomeia um arquivo
- fflush — "descarrega" um arquivo

2) Declaração de um arquivo:

```
FILE *arquivo;
```

3) Abertura de um arquivo:

```
FILE *fopen (const char* nomearq, const char* modo);
```

Obs.: Se ocorrer algum erro na abertura do arquivo, o ponteiro *NULL* é retornado. Por isso, recomenda-se que seja testado o valor retornado pela função *fopen()* antes de começar a manipular o conteúdo do arquivo.

Modos:

r	Abre um arquivo para leitura
r+	Abre um arquivo para leitura/escrita
w	Cria um arquivo para escrita
w+	Cria um arquivo para escrita/escrita
a	*Append*
a+	*Append* – leitura/escrita

Para manipular aquivos binários, os modos descritos acima devem ser acrescidos da letra *b*, resultando em *rb*, *r+b*, *wb*, *w+b*, *ab*, *a+b*.

4) Fechando o arquivo:

int fclose (FILE arquivo);*

Obs.: Se o valor 0 (zero) é retornado, houve sucesso ao fechar o arquivo; qualquer outro valor significa erro.

5) Armazenando caracteres em um arquivo (*putc()* ou *fputc()*):

int putc (int ch, FILE arquivo);*

Obs.: Se a escrita foi realizada, *ch* é retornado; caso contrário, é retornado *EOF*.

6) Lendo caracteres de um arquivo (*getc()* ou *fgetc()*):

int getc (FILE arquivo);*

Obs.: quando o final do arquivo é atingido, *EOF* é retornado. *EOF* também é retornado no caso da ocorrência de algum erro.

7) Testando o final do arquivo (*feof()*):

int feof (FILE arquivo);*

Obs.: devolve "*verdadeiro*" se o final do arquivo foi atingido.

8) Trabalhando com strings (*fputs()* e *fgets()*):

*int fputs (const char *str, FILE *arquivo);*
//Retorna 0 para sucesso; EOF para fracasso

*char *fgets (char *str, int length, FILE *arquivo);*

9) rewind*()*:

Obs.: coloca o cursor para a posição inicial do arquivo.

*void rewind (FILE *arquivo);*

10) *ferror()*:

*int ferror (FILE *arquivo);*

Obs.: devolve *"verdadeiro"* se ocorreu um erro durante a última operação no arquivo; caso contrário, retorna "*falso*".

11) Apagando um arquivo:

int remove (const char nomearq);*

Obs.: devolve 0 (zero) se a operação for bem sucedida, e um valor diferente de 0 (zero) caso não seja.

12) Renomeando um arquivo:

*int rename(const char *nomeAnterior, const char *novoNome)*

Obs.: devolve 0 (zero) se a operação for bem sucedida ou -1 caso não seja.

13) "Atualizando" um arquivo:

*int fflush (FILE *arquivo);*

Obs.: escreve o conteúdo de qualquer dado existente no buffer para o arquivo especificado. Se for passado um valor nulo para *fflush()*, todos os arquivos abertos para saída serão "atualizados".

14) *fread()* e *fwrite()*:

*size_t fread (void *buffer, size_t num_bytes, size_t count, FILE *arquivo);*

*size_t fwrite (const void *buffer, size_t num_bytes, size_t count, FILE *arquivo);*

- *num_bytes: número de bytes a ler ou escrever.*
- *count: determina quantos itens (de num_bytes bytes) serão lidos ou escritos.*

Obs.: a função *fread()* devolve o número de itens lidos, valor este que pode ser menor do que *count*, caso o final do arquivo seja atingido ou ocorra algum erro. A função *fwrite()* devolve o número de itens escritos, que será igual a *count*, a menos que ocorra algum tipo de erro.

O uso dessas funções se mostra mais eficiente no uso de registros (*structs*), pois não se faz necessário especificar o campo sendo lido ou escrito.

15) *fseek()*:

*int fseek (FILE *arquivo, long num_bytes, int origin);*

- *num_bytes: número de bytes a partir de **origin**, que se tornará a nova posição corrente.*
- *origin: pode assumir três valores diferentes: i) SEEK_SET – início do arquivo; ii) SEEK_CUR – posição atual; e iii) SEEK_END – final do arquivo.*

Obs.: a função *fseek()* retorna 0 (zero) quando bem sucedida e um valor diferente de 0 (zero) caso ocorra algum erro.

16) *fprintf()* e *fscanf()*:

*int fprintf (FILE *arquivo, const char *control_string, ...);*

*int fscanf (FILE *arquivo, const char *control_string, ...);*