

# Arquivo Texto

Algoritmos e Programação 2

Prof. Dr. Anderson Bessa da Costa

Universidade Federal de Mato Grosso do Sul

# Introdução

- Estruturas de dados manipuladas fora do ambiente do programa são conhecidas como **arquivos**
- Ambiente do programa é a memória principal
  - Nem sempre é possível armazenar
  - Nem sempre conveniente manter certas estruturas de dados
    - Memória principal é volátil

# Arquivos em C

- **Arquivo em C** representa diversas coisas
  - Arquivos em disco
  - Impressora
  - Teclado ou
  - qualquer dispositivo de entrada e saída
- Esta aula considera apenas **arquivos** em disco

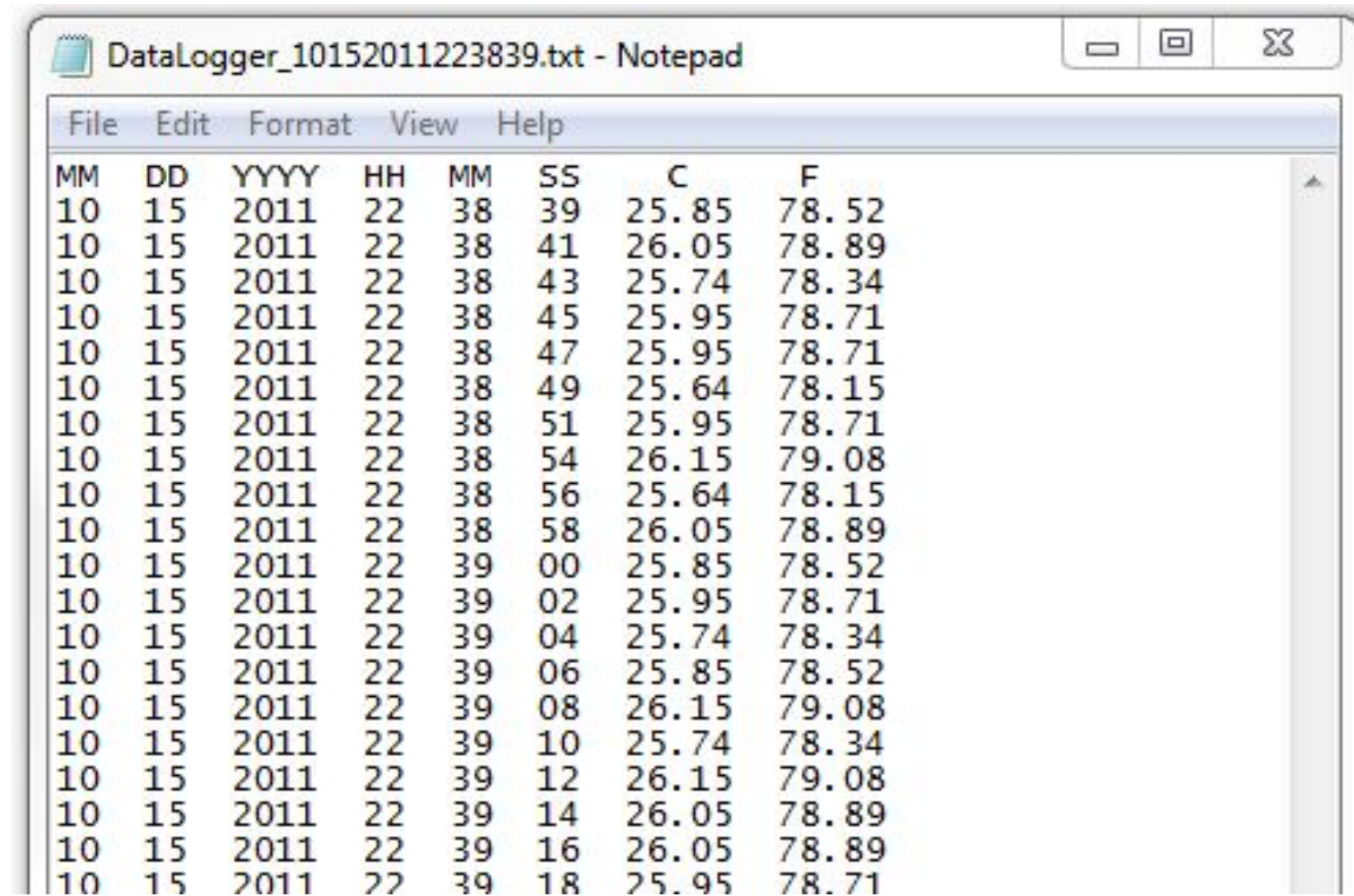
# Biblioteca stdio.h

- C dá suporte à utilização de arquivos c/ biblioteca **stdio.h**
- Fornece funções para manipulação de arquivos, define novos tipos de dados, como o tipo **FILE**
- Uma variável do tipo **FILE** é capaz de identificar um arquivo no disco, direcionando para ele todas as operações

# Tipos de Arquivos em C

- Duas formas de se gravar os dados em um arquivo em C:
  - **Arquivos texto:** podem ser lidos diretamente por qualquer editor de texto
  - **Arquivos binário:** devem ser lidos por programas especiais que convertem a cadeia de bits em informação compreensível

# Arquivo Texto no Notepad



MM	DD	YYYY	HH	MM	SS	C	F
10	15	2011	22	38	39	25.85	78.52
10	15	2011	22	38	41	26.05	78.89
10	15	2011	22	38	43	25.74	78.34
10	15	2011	22	38	45	25.95	78.71
10	15	2011	22	38	47	25.95	78.71
10	15	2011	22	38	49	25.64	78.15
10	15	2011	22	38	51	25.95	78.71
10	15	2011	22	38	54	26.15	79.08
10	15	2011	22	38	56	25.64	78.15
10	15	2011	22	38	58	26.05	78.89
10	15	2011	22	39	00	25.85	78.52
10	15	2011	22	39	02	25.95	78.71
10	15	2011	22	39	04	25.74	78.34
10	15	2011	22	39	06	25.85	78.52
10	15	2011	22	39	08	26.15	79.08
10	15	2011	22	39	10	25.74	78.34
10	15	2011	22	39	12	26.15	79.08
10	15	2011	22	39	14	26.05	78.89
10	15	2011	22	39	16	26.05	78.89
10	15	2011	22	39	18	25.95	78.71

Cada byte representa um caractere (no caso de arquivo no formato ASCII).



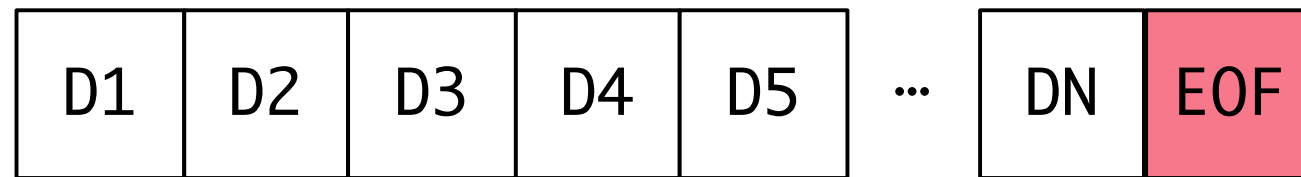
# Arquivo Binário no Notepad

ÿØÿáExifl\*ÿiDucky<ÿáhttp://ns.adobe.com/xap/1.0/ ÿiAdobedÄÿÜ ÿÄ±±ÿÄ@ !1AQaq"2i±BF  
!G8\*[J78ÒcmòÐdò%MiZOðálo6@'ó\$ã"Ólyhk"s!\$r t@9ñ8Í³BÖ«#nãããÄØZçÖùA3L°c&. 8:NDÖ  
áÄHygB°Tê"IMi°¥Ga# {P+CÄ»äliÄWO@bNgÉääÖÖÆIZÖI#ÿ1dY"éiÆ@+¬1Tê«piÄ±i" é¬Js  
0BRa¶Y/A\_Mý2{8ã}2-zpó'Up~nīmīYfrC#ð Êæ7Z+ógL+Ö¬33(ñ|DZHóf£#AÄN+ àFÆ" Jp ôj; 3  
³cP°q%"HdÜiPÖ@Äð2#ÖXyøIÿY\*ÊÜknßLKlājÖ"ð|~>è½kéÇ"CV¿>8\_@çÿl\_öÿjØ6~m\$NQLp|iz  
Ty-é9«h,ã@ãÄöeÜ ÍúÊwP¶ÿÿÜ¶[Í¼¤ÐQ±Ç}i³Pñ]\*önòÄþéZSÜñÄý½NncÑu=ðç¬»Ó¬r({xaÖ@ëZI"  
";k+W¶wiE½ÖbdjóÄjãñVÖú?\_m\_Y³¼U4ãØÄ+ékgðV\ÜZÉ#4¿·Tô&q\_Ê×ÄV#?1äçtkàsÑ^9áá@  
ñb&`@FJ±g\$æU¥i0ãÜz'D.b {±Çj²b¶¶T9ß`HHs9ÉêÆL°, %hsíÄ:Çései©ÊÄK1@§0F0½\$ijj¥4Ö|  
Üð:#A'MIÄ&'CP«ç¥A@××Ç 2:-Üé&¥|p:vçl5\$ý, ^Ä\$ðl« ðyâbðha+¼ÄHÆ.Ç½b'<,wçlKçÆ|q¶HVt  
Gpø¼±ðán°F \*N@pà¾fz4¼"J§[ñÑÉ)äÄÄèðlþjÄÊüä{Í%mqÄ"Ê«ÿ :=;m%(%U@kZ«)S\k }hãNÜ  
\_\*@äi"}Um\$Mg¾Ä"Üú(¥Z2ä\_áÜsÖ×\_ÿik9¾¶Öayl-YH+æâ2áCù+NFeB0ðl 1-<ÄÇ 0\*h)3lÿxEv  
©Ö)ÉÜÐ+S@scj:@.¼üþ3a à9á²-Z'ÎÊ"q\$E@ät·ð Ê"Vçª) Qæ9X2øKxdÐ¥x;|pA\*9oVrLi@i©Ä<  
Sov°z|ÄRµâ|0Q°ÖÄsÄ@ikjâ"Ä\*7Ö9ÄR\_VaÖi ?Üh³1ÜH¾z>f g^iâ nµ¾Ëµj^ÜñÜ«9^,rÆð×&¼il  
lîÉÄ¶ò2k3èø°ßc+)jü=ð½µ,0 AÜ9mªÉÉx\_ zwÜÓsis6µTáð0Ug+"Yu-ó)Äé[bê,M#V\$S\$!8öpÄ½X  
B·3lXlÉÜ|Qlp üÿ«½à(l°Ö u þ²xäirUNð5ç+Lo¶ÆêWq Y)KpCgè35-M'ErJUËc Ê'ÄÖÖmKZ\$mJ£:  
+`R`mýéÖLèÄfñ©Üê«àLó\_ ^mPönÜâ ëjÿl/ijñGjÄK|/Mu£Ä»ð°Ü@lÉ'Ü/lop¼j;f%·Ld|vÄ-Ü[C >¶Ö  
jÊðÄÆÖÖfÜ×H\Ü»Ð cZ+`[Pk!úç§÷Yn»O§Ü!äñðÖCùÄUðBÊ+éae@¼h°ÿÓÿC-iy§'lç\_ÊÜðQpDN  
ùO½ÿ06)|Ði6+r",¿U( Pls=ßzþä§ÿÿ\_§\_ ^ÿm%YäñÆóÆVÝPRµ8é~UN¶@SMÝ¼ØÜtäµz"/- ^8\  
ð4ç|ÜÄ³L¼PAIäc'DVNtçðèl>Ü°Krä"Uìè>ÖþCðA°¾3Qä9eÜi\*·èCnþ@ù¾P258¥5¶ÄçyeÖh\$Z?è  
Öl3Élþ<Ö3w¶l~<«§±Ö¥LKÝl\_ :Äx66lél6ði<@µ·Xü\* &foVç8UÖS¥;±è¥ÖæiRmèðnVFJ»l+5f zÉÜ  
ÄÄçYØQ@®EpUwêb\_Ê¬ÖÄØuQè¶© Ngú°D?Êæÿ\_ÿÓvè2U«ÜQòsUä)Fil4NüØi«ÊÆ)TE³VB  
JApJCéj-ú{ +l±zÄ@\_ñÆêLÉT5\$ÖPr 2,6U|QÆ|>8lP7°t4×0zvÖ l-@µl xWUÖäØuN¾¾ämyln-xE;  
ëjÖ-Ä,A|DÖ¬r5ÊjÄQKY%í½@µÓquÜ2²Ä±GD?6t@xlià%öywYÄ{ð»eÄ²Ü'pT ©(\$H:¥qÇèó+£N¿  
l(>oß¥¬Äl3<@ççlK²lðEná¼{ ØHd1\_"\_ü± ©AÜ<F^PnYé¹ÜlÖÖèe>»ç0ødµçL|Çp³é|tl{§Ö/g·ÿè¶  
/Êøc½¬³® Ço@LÉ+ÊñÄ%6^t+Tny=m6lîläUMQäñ·äg·ÄJlµrSµýóÜ§-zçÖ8Ö[r""h²^a~ð+ÿ\_ìvè  
\_lîÊ"x`Ê8ç@ÄqÄ¹»BälpF Wð§ÜU\$Eit\_Ê'a^Ä7Bjtpä-)lÄ¶cQ ¥g²\_Ü2Ö;ÜL±lµ?"æj3ä{üÐi²"ZØeA  
çÄZwä×¥ÖÖ}ñ¾jA1m|PK F?7\_äNAYTèÿ÷"m,mïöRi<²GÆVéWSTñÖ°4'pG-·g'F{¼¹³>ço¹Ü'Êma§  
Ü;4y\*òJp»"ÿÜ½·wß:&Üxê¹äðGxt\*Gü#%çRa7h¬\_eYÓsz\ZÉ7)'Ü°ÖÜwPjy"iüF%Z'ðègI¾èn¾lÜ  
¥OV9lçjÈcL HØ.ÜlâÆi)|@lâQjç`=9æYWLyix'[Ö;>äg°óí \$(çÉBl' z6lNÜS3(6ØvèX£¾iäÜa¶U«  
{KSWÇ¿NZ/ivçÖ± u&ðÉ'Í»Ü6«'¶¶lù³5ükF§°ëüG»rÖXü0ððl+ü¥¶lóc^x+ðSç<+iibÓ§¶+BU%V;  
hêáÄT:çUrlÄ¶#PÚ/(lÖöÜÿ^á>\*pcb·rðDðm6;D§ H\$?&¬:mÉ1RÆ ¼Hl\*F8+SYBC ¾jµs"8\$Çñ%  
aRäÄW³¿ Ü@yo'Vb« U\Pr LH7Q§Ö"-R"jrøpÄ\$ÄID í¬6|Äç+c±ÄCAL3d;Ê§zÉZqÄ¾l§K@\_Ü³#l  
+CWCj5çäWÖ¿\$iä±ÆèùcQèYfi8oY'/lil'Þ"2\_jæpkèÜÖÜ;±fDüä³ir^°B»¶@æS ¿§+9.ÿ¹FàUU@ð  
D\_JÉN9-RIP ÖW!èXCÉlÖ#lüh8)æjÿjÿv¶¶j£1lÖ¶#Ä@|Ä þG¶lÄjéány²#1i±°b æ9Ö+`ð1 lã?9Èp©Ç  
ç<½Üíÿ òüëiNð¥¼wKG^Üä"py/:=K\$:éÜNl6½ù:k¥rPl×uð\_pÜoißmjlíF5JáEjjZæ1Öiòri@l\_päYN  
ñ[Ö5a!\_ÊµäBpi\* «1VÝQm³ixYbKðð"fldÊ¬\$4Ö/÷ßULÜ(zl¶¶)ÖRYTW×DVÈNV" lri5ÉÉÿà' Øu|GÖ  
ÄÖ"æÄ :3u0f/ðDÉ7ðçfW?xðl¬@YÄHph%¿.F²5w.ÄC¾ðlù¾¼4+ÄlHlâwFlllÜX1äpY7ÄlâÄ·ð²h

Exemplo de quando se tenta abrir um arquivo binário como texto.

# O que é um arquivo?

- Um arquivo é uma sequência de bytes (D1, D2, D3, D4, D5, ... DN)



- Terminada por uma marca de fim de arquivo (**EOF**).



# Abrir Arquivo

A função **fopen()** abre um arquivo, retornando o ponteiro associado a esse arquivo

```
FILE *p;  
p = fopen("nome_do_arquivo", "modo_de_abertura");
```

onde:

- **p:** armazenará o endereço inicial de memória ocupado por um arquivo (se, por qualquer motivo, o arquivo não puder ser aberto, a variável p receberá o valor **NULL**).
- **nome\_do\_arquivo:** é o nome do arquivo que se deseja abrir.
- **modo\_de\_abertura:** representa o modo como arquivo será aberto.

# Modos Abertura Arquivo em C

Modo	Descrição
r	Abre um arquivo de texto onde poderão ser realizadas apenas leituras.
w	Cria um arquivo de texto onde poderão ser realizadas apenas operações de escrita.
a	Anexa novos dados a um arquivo de texto.
rb	Abre um arquivo binário onde poderão ser realizadas apenas leituras.
wb	Cria um arquivo binário onde poderão ser realizadas apenas operações de escrita.
ab	Anexa novos dados a um arquivo binário.
r+	Abre um arquivo de texto onde poderão ser realizadas operações de leitura e de escrita.
w+	Cria um arquivo de texto onde poderão ser realizadas operações de leitura e de escrita.
a+	Anexa novos dados ou cria um arquivo de texto para operações de leitura e escrita.
rb+	Abre um arquivo binário onde poderão ser realizadas operações de leitura e de escrita.
wb+	Cria um arquivo binário onde poderão ser realizadas operações de leitura e de escrita.
ab+	Anexa novos dados a um arquivo binário para operações de leitura e de escrita.

Modos possíveis de abrir um arquivo em C.

# Exemplo 1

```
#include <stdio.h>

int main() {
    // declare um ponteiro para arquivo
    FILE *arq;

    // abra um arquivo no modo escrita
    arq = fopen("arquivo.txt", "w");

    return 0;
}
```

# Abrir Arquivo: Modo escrita

- Quando a função *fopen()* é utilizada para abrir um arquivo no modo escrita (*w* e *wb*), duas situações podem ocorrer:
  1. Se arquivo **não existir**, cria-se;
  2. Se o arquivo **já existir**, sobrepõe-se por um novo arquivo vazio.

# Abrir Arquivo: Modo escrita (cont.)

- Se **fopen()** for executada sem problemas, variável **arq** receberá o endereço de memória ocupado pelo arquivo
- Caso algum erro ocorra, a variável **arq** receberá o valor *NULL*.
  - Altamente recomendável teste para verificar se o arquivo foi aberto adequadamente



# Exemplo 2

```
#include <stdio.h>

int main () {
    FILE *arq;
    arq = fopen("arquivo.txt", "r");

    // Verifique se arquivo foi aberto corretamente
    if(arq == NULL) {
        // Provavelmente arquivo nao existe
        printf("\nOcorreu um erro. O arquivo nao foi aberto.");
    }
    else {
        /* As demais instruções do programa só poderão ser executadas
        se o arquivo foi aberto corretamente */
    }
    return 0;
}
```

# Abrir Arquivo: Modo leitura

- Anteriormente, função **fopen()** foi utilizada para abrir um arquivo existente no modo leitura ("**r**"). Duas situações possíveis:
  1. Se arquivo existe, variável **arq** receberá seu endereço;
  2. Se arquivo não existe, variável **arq** receberá *NULL*.

# Fechar Arquivo

- Função **fclose()** fecha um arquivo
- Após fazer todo o uso necessário do arquivo, é preciso fechá-lo

# Fechar Arquivo (cont.)

- Há dois bons motivos para fazer isso:
  1. Liberamos recursos para o sistema operacional.
  2. Fazemos com que informações não salvas fisicamente no arquivo sejam efetivamente escritas nele, evitando problemas no caso de falhas, travamento ou quedas de energia.

# Exemplo 3

```
#include <stdio.h>

int main () {
    FILE *arq;
    arq = fopen("arquivo.txt", "w");

    if(arq == NULL) {
        printf("\nErro na abertura.");
    }
    else {
        printf("\nSucesso na abertura.");

        /* As demais instruções do programa só poderão ser executadas
        se o arquivo foi aberto de forma correta */

        fclose(arq);
    }
    return 0;
}
```



Arquivos Texto

# Ler e Escrever em Arquivo Texto

- Comando **fprintf** funciona de forma similar ao **printf**, porém com a diferença que escrevemos no arquivo ao invés da tela
- Comando **fscanf** funciona de forma similar ao **fscanf**, porém com a diferença que lemos do arquivo ao invés do teclado

# Exemplo 4

Faça um programa para gerar 100 números aleatórios no intervalo de 1-100 e escreva estes números no arquivo *100\_aleatorios.txt*.

# Exemplo 4: Solução

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

int main() {
    int i, r;
    FILE *f;

    // abra arquivo
    f = fopen("100_aleatorios.txt", "w");
    if (f == NULL) {
        puts("Erro ao abrir o arquivo!");
        exit(1);
    }

    srand(time(NULL));
    for (i = 0; i < 100; i++) {
        // gere um numero aleatorio no intervalo 1 a 100
        r = (rand() % 100) + 1;

        // escreva o numero no arquivo
        fprintf(f, "%d ", r);
    }
    fclose(f);
    return 0;
}
```

# Exemplo 5

Leia o arquivo *100\_aleatorios.txt*, e encontre o maior número gerado. Imprima na tela esse número.



# Exemplo 5: Solução

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

int main() {
    int i, maior, r;
    FILE* f;

    // abra o arquivo
    f = fopen("100_aleatorios.txt", "r");
    if (f == NULL) {
        puts("Erro ao abrir o arquivo!");
        exit(1);
    }
    maior = 0;
    for (i = 0; i < 100; i++) {
        // leia um inteiro do arquivo
        fscanf(f, "%d", &r);

        // se inteiro lido foi maior que o maior inteiro ate entao
        if (r > maior)
            maior = r;
    }
    printf("%d\n", maior);
    fclose(f);
    return 0;
}
```

# Exemplo 6

Imagine agora o seguinte cenário: você deve abrir e ler um arquivo que possui números inteiros espaçados e imprimir na tela todos os números. Porém, você não sabe a priori quantos números existem salvos no arquivo.

# Função feof

## feof

<stdio>

```
int feof ( FILE * stream );
```

### Check end-of-file indicator

Checks whether the *end-of-File indicator* associated with *stream* is set, returning a value different from zero if it is.

This indicator is generally set by a previous operation on the *stream* that attempted to read at or past the end-of-file.

Notice that *stream*'s internal position indicator may point to the *end-of-file* for the next operation, but still, the *end-of-file* indicator may not be set until an operation attempts to read at that point.

This indicator is cleared by a call to [clearerr](#), [rewind](#), [fseek](#), [fsetpos](#) or [freopen](#). Although if the *position indicator* is not repositioned by such a call, the next i/o operation is likely to set the indicator again.

# Exemplo 6: Solução

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

int main() {
    int r, nread;
    FILE *f;

    // abra o arquivo
    f = fopen("entrada.txt", "r");
    if (f == NULL) {
        puts("Erro ao abrir o arquivo!");
        exit(1);
    }
    // enquanto nao chegou no final do arquivo
    while (!feof(f)) {
        // leia um inteiro do arquivo
        nread = fscanf(f, "%d ", &r);

        // se nada foi lido do arquivo, finalize-o
        if(nread == 0)
            break;
        printf("%d ", r);
    }
    fclose(f);
    return 0;
}
```

## Atenção

Existe uma diferença que pode ocorrer no Windows, Linux ou Mac dependendo do compilador utilizado.

Em alguns compiladores basta o **ponteiro** estar no fim do arquivo para ativar o **feof**. Em outros casos é requerido que além do **ponteiro** estar no final, tente-se ler um dado (**fscanf**) para ativar o **feof**.



# Referências

- ASCENCIO, A. F. G.; CAMPOS, E. A. V. Fundamentos da programação de computadores: algoritmos, Pascal, C/C++ e Java. 3. ed. São Paulo: Pearson, 2012.
- DEITEL, P.; DEITEL, H. C: Como Programar. 6<sup>a</sup> ed. São Paulo: Pearson, 2011.
- PIVA, D.J. et al. Algoritmos e programação de computadores. Rio de Janeiro: Elsevier, 2012.