

# Algoritmos e Estruturas de Dados I

## Manipulação de Arquivos Texto

Prof. Ivre Marjorie

# Introdução

---

- ▶ A palavra arquivo será usada para indicar um ‘fluxo de bytes’ (stream)
- ▶ Todo lugar que tem capacidade para receber bytes da memória do computador ou transferi-los para ela recebe o nome de arquivo
- ▶ Exemplos:
  - ▶ arquivos em disco, teclado, vídeo, impressora, portas de comunicação, etc.
- ▶ Vamos trabalhar aqui apenas com arquivos texto salvos em disco



# Introdução

---

- ▶ Em C, as informações necessárias são guardadas em uma estrutura do tipo FILE
- ▶ A definição dessa estrutura está contida no arquivo `stdio.h`
- ▶ O programador não precisa se preocupar com o conteúdo dos membros da estrutura FILE, e sim em como trabalhar com ele, ou seja, abertura, leitura e escrita no arquivo



# Introdução

---

- ▶ Para trabalhar com arquivos, a linguagem C oferece um pacote de funções de biblioteca divididas em **4 grupos**:
  - ▶ Grupo1: Gravar e escrever um caractere por vez (funções `fputc()` e `fgetc()`)
  - ▶ Grupo2: Ler e gravar linha a linha (funções `fputs()` e `fgets()`)
  - ▶ Grupo3: Ler e gravar dados formatados (funções `fprintf()` e `fscanf()`)
  - ▶ Grupo4: Ler e gravar blocos de bytes (funções `fwrite()` e `fread()`)



# Abrindo arquivos

---

- ▶ A função para abrir arquivos é a `fopen()`
- ▶ Essa função executa duas tarefas:
  - ▶ Cria e preenche uma estrutura `FILE`, com as informações necessárias
  - ▶ Retorna um ponteiro do tipo `FILE` que aponta para a localização na memória dessa estrutura criada
- ▶ A função `fopen()` abre um arquivo, retornando o ponteiro associado ao arquivo

```
FILE *fopen (nome_arquivo, modo_abertura);
```

**nome\_arquivo:** representa o nome do arquivo que se deseja abrir, podendo conter inclusive o caminho.

**modo\_abertura:** representa como o arquivo será aberto.

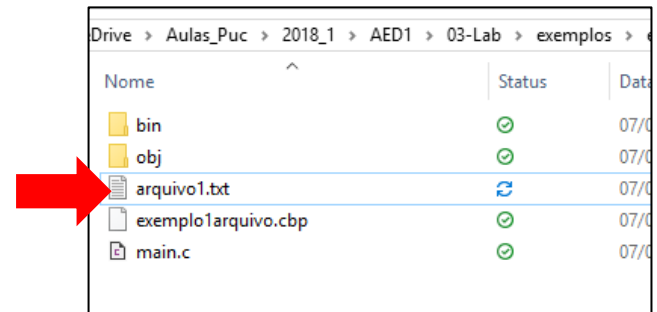


# Exemplo 1

- ▶ Programa que cria o arquivo l.txt para escrita no mesmo diretório em que o projeto está sendo executado.

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

int main()
{
    FILE *arquivo;
    arquivo = fopen("arquivo l.txt","w");
    printf("Arquivo criado.");
    return 0;
}
```



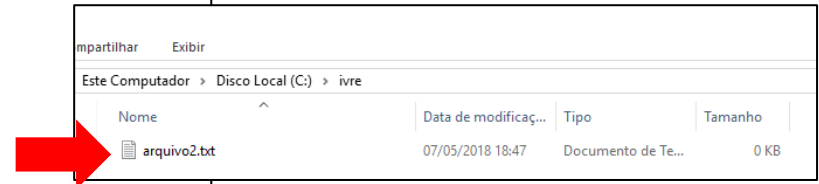


## Exemplo 2

- ▶ Programa que cria o arquivo2.txt para escrita no diretório c:/ivre/

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

int main()
{
    FILE *arquivo;
    arquivo = fopen("c:/ivre/arquivo2.txt","w");
    printf("Arquivo criado.");
    return 0;
}
```



# Modos de abertura de arquivos

Tabela com os possíveis modos de abertura de arquivos:	
"r"	Abre um arquivo para leitura em modo texto. Se o arquivo não existir, a operação irá falhar e <b>fopen()</b> retornará NULL.
"w"	Cria um arquivo em modo texto para gravação. Se o arquivo já existir, elimina seu conteúdo e recomeça a gravação a partir de seu início.
"a"	Abre um arquivo em modo texto para gravação, a partir de seu final. Se o arquivo não existir, ele será criado.
"r+"	Abre um arquivo em modo texto para atualização, ou seja, tanto para leitura como para gravação. Se o arquivo não existir, a operação irá falhar e <b>fopen()</b> retornará NULL.
"w+"	Cria um arquivo em modo texto para atualização, ou seja, tanto para leitura como para gravação. Se o arquivo já existir, seu conteúdo será destruído.
"a+"	Abre um arquivo em modo texto para atualização, gravando novos dados a partir do final do arquivo. Se o arquivo não existir, ele será criado.
"rb"	Abre um arquivo para leitura em modo binário. Se o arquivo não existir, a operação irá falhar e <b>fopen()</b> retornará NULL.
"wb"	Cria um arquivo em modo binário para gravação. Se o arquivo já existir, elimina seu conteúdo e recomeça a gravação a partir de seu início.
"ab"	Abre um arquivo em modo binário para gravação, a partir de seu final. Se o arquivo não existir, será criado.
"rb+"	Abre um arquivo em modo binário para atualização, ou seja, tanto para leitura como para gravação. Se o arquivo não existir, a operação irá falhar e <b>fopen()</b> retornará NULL.



## Grupo1: Gravar e escrever um caractere por vez

---

- ▶ As funções `fputc()` e `fgetc()` gravam e leem um caractere por vez
- ▶ A função `fputc()` escreve um caractere em um arquivo

```
int fputc (char ch, FILE *arq);
```

- ▶ A função `fputc()` recebe dois argumentos: o caractere a ser gravado e o ponteiro para a estrutura `FILE` do arquivo
- ▶ Essa função retorna o caractere gravado ou EOF se acontecer algum erro



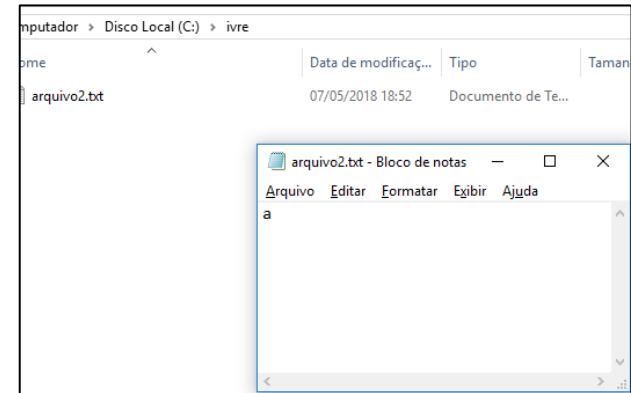


## Exemplo 3

- ▶ Abre um arquivo para anexar novos dados e escreve o caracter a.

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

int main()
{
    FILE *arquivo;
    arquivo = fopen("c:/ivre/arquivo2.txt","a");
    fputc('a',arquivo);
    printf("Escreveu a no arquivo.");
    return 0;
}
```



# Função fclose()

---

- ▶ Quando acabamos de usar um arquivo que abrimos, devemos fecha-lo
- ▶ Para fechar um arquivo devemos usar a função fclose()
- ▶ Fechar um arquivo significa esvaziar o seu *buffer*, gravando nele os caracteres remanescentes

```
int fclose (FILE *arq);
```

O ponteiro **arq** passado à função **fclose()** determina o arquivo a ser fechado. A função retorna zero no caso de sucesso.



## Exemplo 4

- ▶ Escreve a palavra FIM, caracter por caracter, no arquivo “arquivo3.txt”

```
int main()
{
    FILE *fp;
    fp = fopen("arquivo3.txt","w");
    fputc('F', fp); // Grava o caractere F
    fputc('I', fp); // Grava o caractere I
    fputc('M', fp); // Grava o caractere M
    fputc('\n', fp); // Grava o caractere fim de linha
    fclose(fp);
    printf("\nFim do programa\n");
    return 0;
}
```



## Exemplo 5

---

- ▶ Fecha um arquivo que foi aberto.

```
int main()
{
    FILE *arquivo;
    arquivo = fopen("c:/ivre/arquivo2.txt","r");
    if (fclose(arquivo)==0)
        printf("Fechou o arquivo.");
    else
        printf("Nao fechou o arquivo.");
    return 0;
}
```



## Grupo1: Gravar e escrever um caractere por vez

---

- ▶ A função `fgetc()` faz a leitura de um caractere num arquivo

```
int fgetc (FILE *f);
```

- ▶ A função `fgetc()` recebe como argumento o ponteiro para a estrutura `FILE` do arquivo
- ▶ Retorna o caractere lido ou `EOF` se encontrar o fim do arquivo



# Função feof()

---

- ▶ Verifica se um arquivo chegou ao fim

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

- ▶ Retorna não-zero se o arquivo chegou ao EOF (fim de arquivo), caso contrário retorna zero.
- ▶ Outra forma de se verificar se o final do arquivo foi atingido é comparar o caractere lido por fgetc com EOF.





## Exemplo 6

```
int main()
{
    FILE* arquivo = fopen("c:/ivre/arquivo2.txt", "r"); /* Arquivo para leitura */
    char c;
    int cont=0;
    if(arquivo == NULL)
    {
        printf("Erro na abertura do arquivo");
        system("pause");
        exit(1);
    }
    while (!feof(arquivo)) /* Enquanto não chegar ao final do arquivo */
    {
        c = fgetc(arquivo);
        cont++;
        printf("%c",c); /* imprime o caractere lido */
    }
    printf("\n\nA quantidade de caracteres: %d",cont);
    fclose(arquivo);
    return 0;
}
```

Não esqueça de colocar o arquivo no caminho apontado ou na pasta padrão, pois o arquivo será aberto para leitura





## Exemplo 7

O próximo exemplo lê, do arquivo, um caractere por vez e o imprime no vídeo.

```
/* ifilech.c */
/* Lê um caractere por vez de um arquivo */
#include <stdio.h> /* Define FILE */
#include <stdlib.h>

int main(void)
{
    FILE *fptr; /* Ponteiro para arquivo */
    short int ch;
    /* Abre arquivo para ler em modo texto */
    fptr = fopen("ArqText.txt","r");

    while((ch=fgetc(fptr)) != EOF) /* Lê um caractere do arquivo */
        printf("%c",ch);          /* Imprime o caractere no vídeo */
    fclose(fptr);

    return 0;
}
```


## Grupo2: Ler e gravar linha a linha (funções fputs() e fgets())

---

- ▶ A função fputs() escreve uma cadeia de caracteres em um arquivo

```
int fputs (char *cadeia, FILE *arq);
```

cadeia: variável  
que armazena  
uma cadeia de  
caracteres



- ▶ A função fputs recebe dois argumentos: o ponteiro char, para a cadeia de caracteres a ser gravada, e o ponteiro para a estrutura FILE do arquivo
  - ▶ Retorna um número positivo ou EOF se acontecer algum erro
-

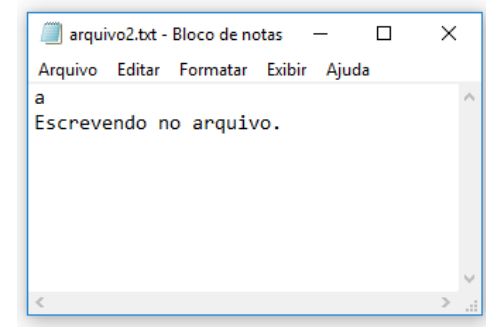


## Exemplo 8

- ▶ Abre um arquivo para anexar dados e escreve a frase “Escrevendo no arquivo.” uma linha abaixo, por causa do `\n`.

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

int main()
{
    FILE *arquivo;
    arquivo = fopen("c:/ivre/arquivo2.txt","a");
    fputs("\nEscrevendo no arquivo.",arquivo);
    printf("Escreveu no arquivo.");
    return 0;
}
```



## Grupo2: Ler e gravar linha a linha (funções fputs() e fgets())

---

- ▶ A função fgets() faz a leitura de uma string contida em um arquivo

```
char *fgets (char *str, int tamanho, FILE *f);
```

- ▶ A função fgets recebe três argumentos: o ponteiro char para a cadeia de caracteres em que os dados lidos do arquivo serão colocados, o número máximo de caracteres a serem lidos, e o ponteiro para a estrutura FILE do arquivo
- ▶ Retorna um ponteiro para a cadeia de caracteres lida ou NULL se encontrar algum erro ou o fim do arquivo



## Grupo2: Ler e gravar linha a linha (funções `fputs()` e `fgets()`)

---

- ▶ A função `fgets` lê a string até que um caractere de nova linha seja lido ou tamanho `-1` caracteres tenham sido lidos
  - ▶ se o caractere de nova linha (`'\n'`) for lido, ele fará parte da string
  - ▶ a string resultante sempre terminará com `'\0'` (por isto somente tamanho-1 caracteres, no máximo, serão lidos)
  - ▶ retorna a própria string, se a leitura for sucedida
  - ▶ retorna `NULL`, caso contrário





## Exemplo 9

---

```
int main(void)
{
    FILE *fptr; /* Ponteiro para arquivo */
    char str[81];

    /*Abre para leitura em modo texto */
    if((fptr = fopen(("TesteSTR.txt","r")) == NULL)
    {
        puts("Não foi possível abrir o arquivo");
        exit(1);
    }

    while(fgets(str,80,fptr) != NULL) /* Lê uma linha de texto */
        printf("%s",str);

    fclose(fptr);
    system("pause");
    return 0;
}
```



## Grupo3: Ler e gravar dados formatados (funções fprintf() e fscanf())

---

- ▶ As funções fprintf() e fscanf() gravam ou leem os dados de modo formatado
- ▶ A função **fprintf()** – escreve no arquivo uma sequência de dados formatados

```
int fprintf ( FILE * f, const char * formato, [argumentos] );
```

- ▶ Retorna a quantidade de caracteres escritos





## Exemplo 10

---

```
/* Grava dados formatados em um arquivo */
#include <stdio.h> /* Define FILE */
#include <stdlib.h>
#include <string.h>
#define TRUE 1

int main(void)
{
    FILE *fptr; /* Ponteiro para arquivo */
    char titulo[30];
    int regnum;
    double preco;
    fptr = fopen("livros.txt","w");
    while (TRUE)
    {
        printf("\nDigite título, registro e preço do livro: ");
        scanf("%s %d %lf",titulo,&regnum,&preco);
        if(strlen(titulo) <= 1) break;
        fprintf(fptr,"%s %d %.2lf\n",titulo,regnum,preco);
    }

    fclose(fptr);
    system("pause");
    return 0;
}
```



## Grupo3: Ler e gravar dados formatados (funções `fprintf()` e `fscanf()`)

---

- ▶ A função **`fscanf()`** faz a leitura do arquivo uma sequência de dados formatados

```
int fscanf ( FILE * f, const char * formato, [argumentos] );
```

- ▶ Retorna a quantidade de itens lidos
- ▶ É similar à função `scanf()`, exceto pelo fato de que, como `fprintf()`, um ponteiro para `FILE` deverá ser incluído como primeiro argumento





# Exemplo 11

---

```
/* Lê dados formatados do arquivo */
#include <stdio.h> /* Define FILE */
#include <stdlib.h>

int main(void)
{
    FILE *fptr; /* Ponteiro para arquivo */
    char titulo[30];
    int regnum;
    double preco;
    fptr = fopen("livros.txt","r");

    while (fscanf(fptr,"%s %d %lf", titulo, &regnum, &preco) != EOF)
        printf("%s %d %.2lf\n", titulo, regnum, preco);

    fclose(fptr);
    system("pause");
    return 0;
}
```



# Função fseek()

---

- ▶ Um ponteiro para um arquivo aponta para um byte particular chamado *posição atual*
- ▶ Todas as funções vistas precisam conhecer o ponteiro para o arquivo que desejamos manipular
- ▶ A cada tempo em que gravamos ou lemos qualquer coisa no arquivo, o ponteiro é movido para o fim dessa coisa e a próxima leitura ou gravação começa nesse ponto
- ▶ Quando o arquivo é aberto, o ponteiro é fixado e, seu primeiro byte; então se quisermos ler ou gravar nele estaremos no seu início



# Função fseek()

---

- ▶ Se abrirmos o arquivo usando a opção/modo de abertura a (append), o seu ponteiro será posicionado no fim do arquivo
- ▶ A função fseek() permite movimentar a posição corrente de leitura e gravação do arquivo para uma posição escolhida

```
int fseek (FILE * fluxo, int deslocamento, int origem);
```



# Função fseek()

---

- ▶ A função `fseek()` aceita três argumentos:
  - ▶ O primeiro é o ponteiro para a estrutura `FILE` do arquivo. Após a chamada dessa função esse ponteiro será movimentado para a posição que desejarmos
  - ▶ O segundo argumento é chamado de *deslocamento*, e consiste no número de bytes que desejamos deslocar a partir da posição especificada pelo terceiro argumento
  - ▶ O terceiro argumento chama-se *posição*. Existem três possibilidades para esse número, as quais determinam de onde o deslocamento começará a ser medido, a saber:

Posição	Significado
0	Início do arquivo
1	Posição atual
2	Fim do arquivo



## Exemplo 12

```
int main()
{
    FILE *arquivo;
    char string[100];
    if((arquivo = fopen("arquivo.txt","w")) == NULL)
    {
        printf("\nErro ao abrir o arquivo ! ");
        exit(1);
    }
    do
    {
        printf("\nDigite uma nova string. Para terminar, aperte somente <enter>: ");
        gets(string);
        fputs(string, arquivo);
        putc('\n', arquivo);
    } while (strlen(string) > 0);
    fclose(arquivo);
    return 0;
}
```



# Exercícios

---

1. Implemente um programa que abra um arquivo texto e conte a quantidade de caracteres 'a' que estão presentes nele. Imprima a quantidade na tela.
2. Implemente um programa que leia um arquivo texto e imprima, linha a linha, o seu conteúdo na tela. Imprima também a quantidade de linhas que este arquivo possui.
3. Faça um programa para inserir os dados (Título e Ano de lançamento) de 10 livros em um arquivo.
4. Faça um programa para inserir 100 letras informadas pelo usuário em um arquivo texto. Depois de inseridas as 100 letras, o programa deverá ler todas as 100 letras do arquivo, calcular e mostrar a quantidade de cada vogal. Usar switch/case para resolução.





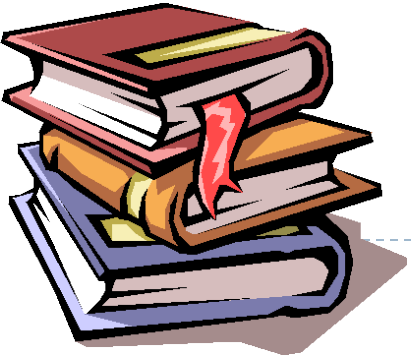
# Exercícios

---

5. Faça um programa para inserir 30 letras informadas pelo usuário em um arquivo texto. Cada letra deverá estar em uma linha diferente do arquivo. Depois de inseridas as 30 letras, o programa deverá ler todas as 30 letras do arquivo, calcular e mostrar a quantidade de letras “c”, “s” e “v”. Usar switch/case para resolução.
6. Faça um programa para inserir os dados (Modelo e Ano de fabricação) de 20 carros em um arquivo.
7. Faça um programa para inserir N letras informadas pelo usuário em um arquivo texto. Onde N é uma quantidade de letras definida pelo usuário. Depois de inseridas as N letras, o programa deverá ler todas as N letras do arquivo, calcular e mostrar a média de letras “a”.







## Referência Bibliográfica

---

- ▶ MIZRAHI, Victorine Viviane. **Treinamento em linguagem C**. São Paulo: Pearson Prentice Hall, 2008. 2ª edição. Curso Completo. Capítulo 11.
- ▶ Site: <http://www.cmaismais.com.br/referencia/cstdio>

