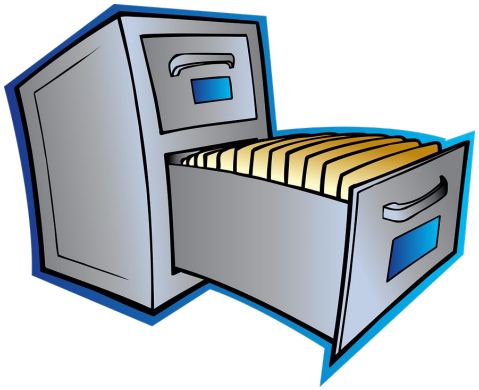
Algoritmos e Programação II

Prof. Joilson dos Reis Brito

Roteiro da Aula

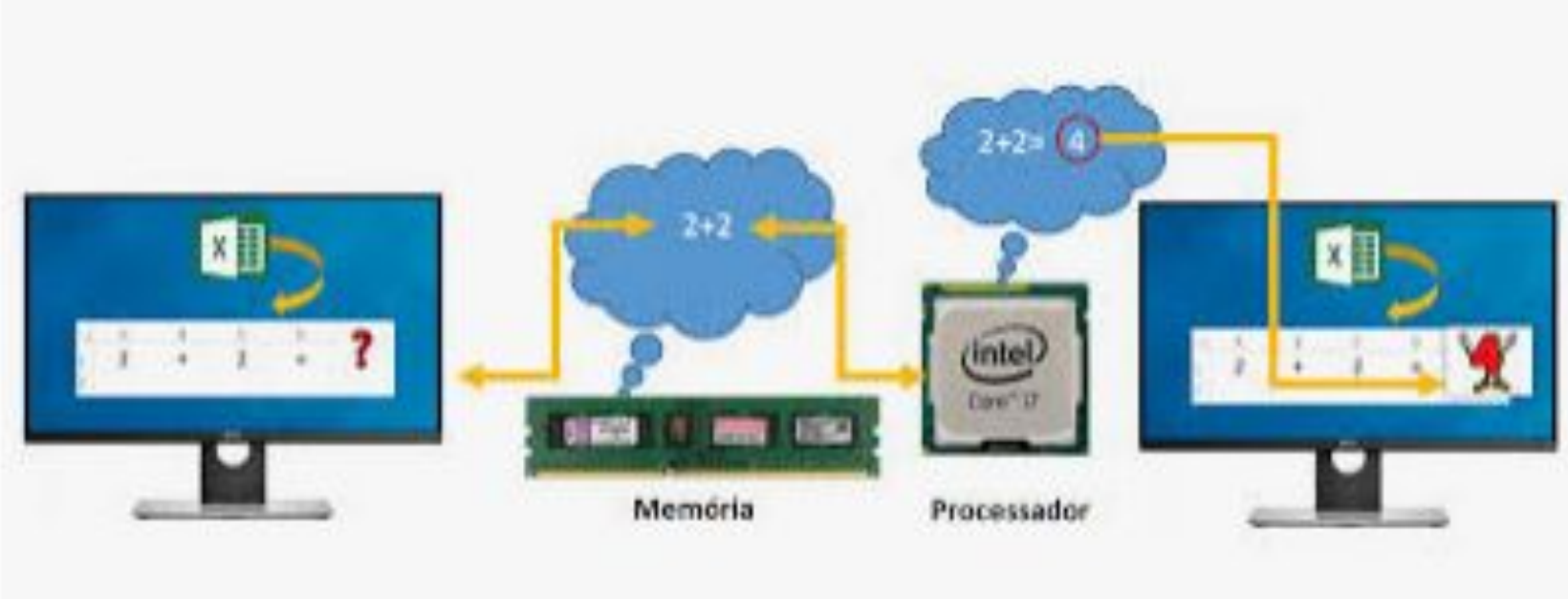
- ✔ Arquivos
- ✔ Exemplos

ARQUIVOS



Até agora, os programas que implementamos solicitam ao usuário que informe os dados que os programas irão manipular.

Uma vez terminado o programa, todos os dados introduzidos, ou mesmo os resultados do programa, eram perdidos, pois eram armazenados na memória RAM do computador, que é uma memória volátil, que se apaga quando o computador é desligado.



Agora vamos estudar sobre **ARQUIVOS,** uma solução para armazenar, em memórias permanentes como HD, SSD, Pen drive e outras unidades secundárias de armazenamento, os dados recebidos ou gerados por nossos programas.

O que são Arquivos?

Arquivos são espaços que permitem o armazenamento de informações, que podem ser: caracteres, palavras, números ou conjunto de informações.

A utilização de arquivos pode ser vista em dois sentidos distintos:

- O arquivo é a fonte de dados para o programa: neste caso trata-se de um arquivo de entrada de dados (*input);*
- O arquivo é o destino dos dados gerados pelo programa: nesse caso, trata-se de um arquivo de saída dos dados (*output).*

Recursos Físicos do Computador

Programas sem utilizar Arquivos

Entrada

Processamento

Saída

Recursos não utilizados

Programa que utilizam Arquivos

Entrada

Processamento

Saída

TIPOS DE ARQUIVOS

Arquivos Texto: Contém apenas os caracteres que formam o texto.

Arquivos Binários: São arquivos gravados por programas que colocam no arquivo, além do seu conteúdo visível, outros códigos de formatação ou de instruções.

Inicialmente trabalharemos com arquivos texto.

Funções para Manipulação de Arquivos

A linguagem C possui uma série de funções para manipulação de arquivos, cujos protótipos estão reunidos na biblioteca padrão de entrada e saída (*Input and Output*), **stdio.h**.

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

Operações Básica sobre Arquivos

- Para processar um arquivo, a primeira operação a ser realizada é ligar uma variável no programa ao arquivo. Chamamos essa operação de **Abertura do Arquivo**.
- A **Abertura do Arquivo** consiste em associar uma variável do nosso programa ao arquivo que pretendemos processar, ou, em outras palavras, representar internamente o nome físico do arquivo através de um nome lógico, que corresponde à variável que irá representá-lo no nosso programa.
- Depois de aberto um arquivo, poderemos realizar todas as operações que pretendermos sobre o conteúdo desse arquivo: ler dados, escrever dados, posicionar ao longo do arquivo para modificar um dado, etc.

Operações Básica sobre Arquivos

As operações de Abertura e Fechamento traduzem-se em inglês por *open e close*.

Em C todas as funções para processamento de arquivos são precedidas pela letra **f** (de *file*), de forma a melhor indicar que se tratam de funções sobre arquivos (como funções que operam strings são precedidas por str - strlen, strcpy).

Funções que permitem abrir e fechar um arquivo em C são **fopen** e **fclose**.

Declaração de Variável para o Processamento de Arquivo

Podemos declarar um ponteiro de arquivo da seguinte maneira:

FILE *PonteiroArquivo;

A declaração da variável do tipo FILE * faz com que PonteiroArquivo seja um ponteiro para o tipo FILE.

O tipo FILE é definido na biblioteca stdio.h.

É usando este tipo de ponteiro que vamos manipular arquivos em C.

Abertura de Arquivo

A abertura de arquivos é realizada usando a função **fopen**, seu protótipo é:

FILE fopen (char *nome_do_arquivo, char *modo_de_abertura);

A função **fopen** recebe dois parâmetros:

nome_do_arquivo - String contendo o nome físico do arquivo (ex: DADOS.txt).

modo_de_abertura - String contendo o modo de abertura do arquivo

Modos de Abertura de Arquivo

Existem três modos de abertura de um arquivo:

MODO	SIGNIFICADO
r	read (Abertura do arquivo para leitura) Abre o arquivo para leitura, caso não consiga abrir, a função fopen devolve NULL (ponteiro Nulo).
w	write (Abertura do arquivo para escrita) Abre o arquivo para escrita. É criado um novo arquivo com o nome passado à função fopen. Se o arquivo já existir, o mesmo será apagado e será criado um arquivo vazio com o nome indicado. caso não consiga abrir, a função fopen devolve NULL.
a	append (Abertura do arquivo para acrescentar) Abre o arquivo para acrescentar. Se o arquivo não existir, e funciona como o modo w. Se o arquivo já existir, coloca-se no final deste, de forma a permitir a escrita dos dados de forma sequencial a partir dos dados já existentes.

Modos de Abertura de Arquivo

Além desses três modos básicos de abertura, é possível também abrir um arquivo de forma a permitir simultaneamente operações de leitura e escrita, colocando o sinal de + após o modo.

MODO	SIGNIFICADO
r+	Abertura do arquivo para **leitura e escrita**. Se o arquivo não existir, é criado. Se o arquivo já existir, os novos dados serão colocados a partir do início do arquivo, por cima, isto é, apagando os dados anteriores.
w+	Abertura do arquivo para **leitura e escrita**. Se o arquivo não existir, é criado. Se o arquivo já existir, é apagado e criado um novo com o mesmo nome.
a+	Abertura do arquivo para **leitura e escrita**. Se o arquivo não existir, é criado. Se o arquivo já existir, os novos dados serão colocados a partir do final do arquivo.

Abertura de Arquivo

Exemplo:

```
FILE *PonteiroArquivo;

PonteiroArquivo = fopen("Dados.txt","r");
```

Nome do Arquivo

O nome do arquivo é armazenado em uma *string* e deve representar fielmente o nome do arquivo tal como é visto pelo sistema operacional que você está utilizando.

Exemplo:

```
FILE *PonteiroArquivo;
PonteiroArquivo = fopen("Dados.txt",”w”);
```

Este comando irá criar o arquivo Dados.txt na pasta corrente onde o código fonte está salvo.

Nome do Arquivo

O nome do arquivo pode estar contido numa **string constante**:

Exemplo:

```
FILE *PonteiroArquivo;
PonteiroArquivo = fopen("Dados.txt","r");
```

Nome do Arquivo

O nome do arquivo também poder estar em uma **variável do tipo string** armazenada na memória.

Exemplo:

```
FILE *PonteiroArquivo;
char NomeArquivo[20];
printf("Digite o nome do arquivo: ");
gets(NomeArquivo);
strcat(NomeArquivo,".txt");
PonteiroArquivo= fopen(NomeArquivo,);
```

Nome do Arquivo

No Windows, caso seja necessário especificar o caminho do arquivo será necessário usar o caracter "\" (C:\LPII\Dados.txt), que é um caracter especial em C.

Por isso, caso o nome do arquivo seja constante, deve-se ter o cuidado de colocar \\ para representar cada caractere \.

Exemplo:

Para utilizar o caminho/arquivo C:\Temp\Dados.txt deveremos escrever:

```
Arq = fopen("C:\\Temp\\Dados.txt","w");
```

Fechamento de Arquivo

O fechamento de um arquivo retira a ligação entre a variável do programa e o arquivo existente no disco.

sintaxe:

int fclose (char *nome_do_arquivo);

A função **fclose** recebe um parâmetro:

nome_do_arquivo - String contendo o nome físico do arquivo (ex: DADOS.txt).

e devolve 0 em caso de sucesso ou a constante EOF (End Of File) em caso de erro.

Realizar o fechamento do arquivo é uma boa prática de programação.

Exemplo de Abertura e Fechamento de Arquivo

Programa que indica se podemos ou não abrir um arquivo.

```
#include<stdio.h>
#include<locale.h>
#include<stdlib.h>
int main()
{
    FILE *PonteiroArquivo;
    char NomeArquivo[100];
    setlocale(LC_ALL,"Portuguese");
    printf("Digite o nome do arquivo: ");
    gets(NomeArquivo);
    PonteiroArquivo = fopen(NomeArquivo,"w");
    if(PonteiroArquivo == NULL)
    {
        printf("Impossível abrir o arquivo %s!\n",NomeArquivo);
    }
    else
    {
        printf("Arquivo %s aberto com sucesso!\n",NomeArquivo);
        fclose(PonteiroArquivo);
    }
    system("Pause");
    return 0;
}
```

Abra a pasta e observe o arquivo criado após salvar e executar o programa.

Abertura de Arquivo

A função **fopen** é responsável pela abertura de arquivos em C.

Se por qualquer razão o arquivo não puder ser aberto, a função devolve NULL, não sendo o programa finalizado automaticamente.

É de responsabilidade do programador o tratamento dos problemas que possam encontrar quando processa arquivos.

Abertura de Arquivo

- Quando o valor NULL for retornado pela função, deve ser possível parar a execução do programa. Para isso, utilizaremos a função **exit**(int num).

- A função exit() permite terminar um programa qualquer que seja o local onde esteja executando. Em geral devolvem-se números diferentes para cada erro, que podem ser aproveitados pelo Sistema Operacional.

- Caso o programa termine sem qualquer problema, pode-se fazer exit(0). Vamos fazer exit(1), porém esse retorno não será usado pelo sistema operacional.

Exemplo 1- Abertura e Fechamento de Arquivo

Programa que indica se podemos ou não abrir um arquivo.

```
#include<stdio.h>
#include<locale.h>
#include<stdlib.h>
int main()
{
    FILE *PonteiroArquivo;
    char NomeArquivo[100];
    setlocale(LC_ALL,"Portuguese");
    printf("Digite o nome e a extensão do arquivo: ");
    gets(NomeArquivo);
    PonteiroArquivo = fopen(NomeArquivo,"w");
    if(PonteiroArquivo == NULL)
    {
        printf("Impossível abrir o arquivo %s!\n",NomeArquivo);
        exit(1);
    }
    else
    {
        printf("Arquivo %s aberto com sucesso!\n",NomeArquivo);
        fclose(PonteiroArquivo);
    }
    system("Pause");
    return 0;
}
```

Abra a pasta e observe o arquivo criado após salvar e executar o programa.

Escrita de caracter em Arquivo

A função **fputc** escreve um caracter em um arquivo que foi previamente aberto para escrita.

Sintaxe:

int fputc(int caracter, FILE *PonteiroArquivo)

A função fputc grava o caracter no arquivo PonteiroArquivo.

Exemplo 2
Escrita de caracteres em Arquivo

```
#include <stdio.h>
#include <stdlib.h>
#include <locale.h>
int main()
{
    FILE *PonteiroArquivo;
    char NomeArquivo[100];
    char Caracter;
    setlocale(LC_ALL,"Portuguese");
    printf("Digite o nome e a extensão do arquivo: ");
    gets(NomeArquivo);
    PonteiroArquivo = fopen(NomeArquivo,"w");
    if(PonteiroArquivo == NULL)
    {
          printf("Ocorreu um erro! O arquivo não foi aberto!\n");
          exit(1);
    }
    else
    {
      printf("\nDigite um caracter: ");
      fflush(stdin);
      scanf("%c",&Caracter);
      while(Caracter != 'f')
      {
           fputc(Caracter,PonteiroArquivo);
           printf("\nDigite outro caracter ou 'f' para sair: ");
           fflush(stdin);
           scanf("%c",&Caracter);
      }
    }
    if (fclose(PonteiroArquivo) == 0)
        printf("\n\nArquivo fechado com sucesso!!!\n\n");
    system("Pause");
    return 0;
}
```

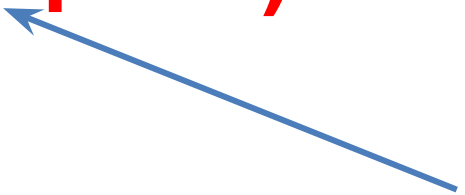
Abra a pasta e observe o arquivo criado após executar o programa.

Leitura de caracter em Arquivo

A função **fgetc** lê um caracter de um arquivo que foi previamente aberto para escrita ou leitura.

Sintaxe:

int fgetc(FILE *PonteiroArquivo)



A função fgetc retorna um caracter lido do arquivo PonteiroArquivo.

Leitura de caracteres em Arquivo

A função fgetc() retorna um int e não um char. Isso se deve ao fato de todos os caracteres da tabela ASCII serem caracteres válidos, podendo todos eles estar dentro de um arquivo.

Como qualquer um dos 256 caracteres da tabela ASCII pode ser lido a partir de um arquivo, a função terá que devolver um símbolo que não seja qualquer um dos caracteres, de forma a indicar que foi detectada uma situação de End Of File.

Isso é conseguido retornando qualquer valor que não esteja entre 0 e 255 (Variação dos códigos ASCII)

A constante simbólica EOF (-1) serve de constante indicadora de End-Of-File.

Exemplo 3 Leitura de caracteres em Arquivo

```
#include <stdio.h>
#include <stdlib.h>
#include <locale.h>
int main()
{
  FILE *PonteiroArquivo;
  char NomeArquivo[100];
  char Caracter;
  setlocale(LC_ALL,"Portuguese");
  printf("Digite o nome do arquivo: ");
  gets(NomeArquivo);
  PonteiroArquivo = fopen(NomeArquivo,"r");
  if(PonteiroArquivo == NULL)
  {
        printf("Ocorreu um erro! O arquivo não foi aberto!\n");
        exit(1);
  }
  else
  {
    printf("\nCaracteres do arquivo %s: \n",NomeArquivo);
    while((Caracter = fgetc(PonteiroArquivo)) != EOF)
    {
       printf("%c",Caracter);
     }
  }
  fclose(PonteiroArquivo);
  system("Pause");
  return 0;
}
```