Material baseado de várias fontes de pesquisa.

**Arquivos em C**

Os arquivos permitem gravar os dados de um programa de forma permanente em mídia digital.

Vantagens de utilizar arquivos.

* **Armazenamento permanente de dados**: as informações permanecem disponíveis mesmo que o programa que as gravou tenha sido encerrado, ou seja, podem ser consultadas a qualquer momento.
* **Grande quantidade dados pode ser armazenada**: A quantidade de dados que pode ser armazenada depende apenas da capacidade disponível da mídia de armazenamento. Normalmente a capacidade da mídia é muito maior do que a capacidade disponível na memória RAM.
* **Acesso concorrente**: Vários programas podem acessar um arquivo de forma concorrente.

A linguagem C trata os arquivos como uma sequência de bytes. Esta sequência pode ser manipulada de várias formas e para tanto, existem funções em C para criar, ler e escrever o conteúdo de arquivos independentemente de quais sejam os dados armazenados.

**Tipos de arquivos**

Em C trabalha-se com dois tipos de arquivos:

1) **Arquivo texto**: Armazena caracteres que podem ser mostrados diretamente na tela ou modificados por um editor de texto.

Exemplos de arquivos texto: documentos de texto, código fonte C, páginas XHTML.

2) **Arquivo binário:** é uma sequência de bits que obedece a regras do programa que o gerou.

Exemplos: Executáveis, documentos do Word, arquivos compactados.

**O ponteiro para arquivo**

Em C, o arquivo é manipulado através de um ponteiro especial para o arquivo.

A função deste ponteiro é “apontar” a localização de um registro.

**Sintaxe: FILE < \*ponteiro >**

O tipo FILE está definido na biblioteca stdio.h.

Exemplo de declaração de um ponteiro para arquivo em C: **FILE \*pont\_arq;**Lembrando que **FILE** deve ser escrito **em letras maiúsculas**.

**Operações com arquivos do tipo texto**

**Abertura de arquivos.**

Para trabalhar com um arquivo, a primeira operação necessária é **abrir este arquivo**.

**Sintaxe de abertura de arquivo**:  
**< ponteiro > = fopen(“nome do arquivo”,”tipo de abertura”);**

A função **fopen** recebe como parâmetros o **nome do arquivo a ser aberto** e **o tipo de abertura** a ser realizado.

Depois de aberto, realizamos as operações necessárias e **fechamos o arquivo**.

Para fechar o arquivo usamos a **função fclose**.

Sintaxe de fechamento de arquivo  
**fclose< ponteiro >;**  
Lembrando que o ponteiro deve ser a mesma variável ponteiro associada ao comando de abertura de arquivo.

**Tipos de abertura de arquivos**

**r:** Permissão de **abertura somente para leitura**. É necessário que o arquivo já esteja presente no disco.

**w:** Permissão de **abertura para escrita (gravação).** Este código cria o arquivo caso ele não exista, e caso o mesmo exista ele recria o arquivo novamente fazendo com que o conteúdo seja perdido. Portanto devemos tomar muito cuidado ao usar esse tipo de abertura.

**a:** Permissão **para abrir um arquivo texto para escrita(gravação), permite acrescentar novos dados ao final do arquivo**. Caso não exista, ele será criado.

**Exemplo de criação de arquivos.**

**#include <stdio.h>**

**#include <stdlib.h>**

**int** main(**void**)

{

FILE \*pont\_arq; // criando a variável ponteiro para o arquivo

pont\_arq = fopen("arquivo.txt", "a"); // abrindo o arquivo

fclose(pont\_arq); // fechando arquivo

printf("O arquivo foi criado com sucesso!"); //mensagem para o usuário

system("pause");

**return**(0);

}

**Problemas na abertura de arquivos**

Na prática, nem sempre é possível abrir um arquivo. Podem ocorrer algumas situações que impedem essa abertura, por exemplo:

* Você está tentando abrir um arquivo no modo de leitura, mas o arquivo não existe;
* Você não tem permissão para ler ou gravar no arquivo;
* O arquivo está bloqueado por estar sendo usado por outro programa.

Quando o arquivo não pode ser aberto a função **fopen** **retorna o valor NULL**.

**Tratando erro ao abrir o arquivo**

É altamente recomendável criar um trecho de código a fim de verificar se a abertura ocorreu com sucesso ou não.

Exemplo:

if (pont\_arq == NULL)

{

printf("ERRO! O arquivo não foi aberto!\n");

}

else

{

printf("O arquivo foi aberto com sucesso!");

}

A função **fprintf** **armazena dados em um arquivo**. Seu funcionamento é muito semelhante ao printf, a **diferença principal é a existência de um parâmetro para informar o arquivo onde os dados serão armazenados.**

**Sintaxe:**

**fprintf(nome\_do\_ponteiro\_para\_o\_arquivo, “%s”,variavel\_string)**

**#include <stdio.h>**

**#include <stdlib.h>**

**int** main(**void**)

{

FILE \*pont\_arq; // cria variável ponteiro para o arquivo

**char** palavra[20]; // variável do tipo string

//abrindo o arquivo com tipo de abertura w

pont\_arq = fopen("arquivo\_palavra.txt", "w");

//testando se o arquivo foi realmente criado

**if**(pont\_arq == **NULL**)

{

printf("Erro na abertura do arquivo!");

**return** 1;

}

printf("Escreva uma palavra para testar gravacao de arquivo: ");

scanf("%s", palavra);

//usando fprintf para armazenar a string no arquivo

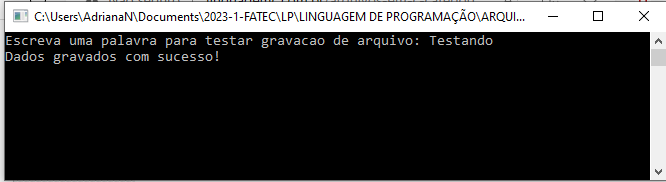
fprintf(pont\_arq, "%s", palavra);

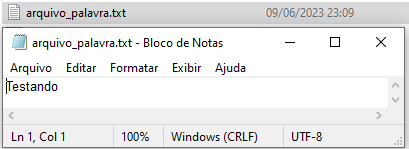
//usando fclose para fechar o arquivo

fclose(pont\_arq);

printf("Dados gravados com sucesso!");

getch();





**return**(0);

}

**Gravando caracter por caracter em um arquivo com fputc**

A função **fputc()** escreve um único caractere em um arquivo.

Na prática temos que **a cada chamada da função fputc grava um caraceter no arquivo**.

Código usando putc() a seguir.

**#include <stdio.h>**

tamanho=strlen(frase);

//gravando caracter por caracter

**for**(i=0; i < tamanho; i++)

{

fputc(frase[i], pont\_arq);

}

fclose(pont\_arq);

**return** 0;

}

**#include <stdlib.h>**

**int** main(**void**)

{

FILE \*pont\_arq;

**char** frase[50];

**int** i;

**int** tamanho;

pont\_arq = fopen("arquivo1.txt","w");

**if** (pont\_arq == **NULL**)

{

printf("Erro ao tentar abrir o arquivo!");

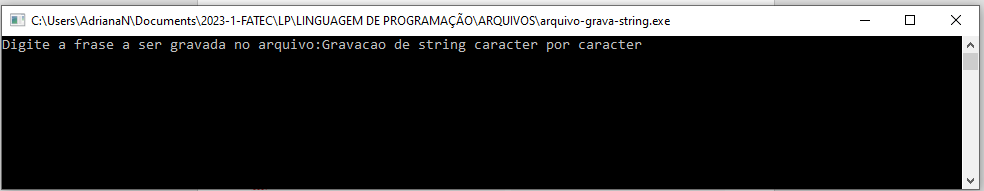
exit(1);

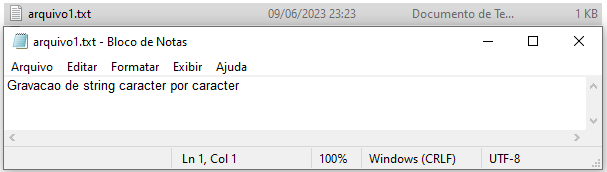
}

printf("Digite a frase a ser gravada no arquivo:");

gets(frase);

//verificando a quantidade de caracteres da string frase





**Função fputs para escrita em arquivos**

A função **fputs()** é usada para **gravar strings em arquivos**.

A função **fputs** recebe como **parâmetro uma string** e **um ponteiro para o arquivo a ser escrito.**

**Retorno da função fputs**:

* Se a **string for gravada com sucesso**, **retorna um inteiro diferente de zero**.
* Caso **ocorra algum erro na escrita**, o **valor da constante EOF é retornado**.

Exemplo:

**#include <stdio.h>**

**#include <stdlib.h>**

**int** main(**void**)

{

FILE \*pont\_arq;

**int** r;

pont\_arq = fopen("arquivo2.txt", "w");

//Testando a abertura do arquivo

**if** (pont\_arq == **NULL**)

{

printf("Erro ao tentar abrir o arquivo!");

exit(1);

}

//Gravando strings no arquivo

r = fputs("Programando em Linguagem C.", pont\_arq);

//Verificando se os dados foram gravados com sucesso

**if**( r == EOF)

{

printf("Erro ao tentar gravar os dados! \n");

}

**else**

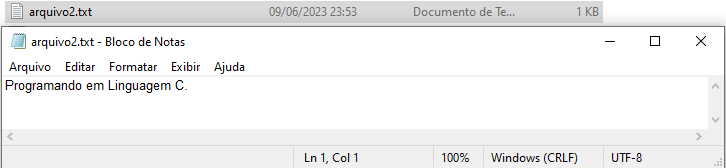
{

printf("Dados gravados com sucesso. \n");

}

Texto

Descrição gerada automaticamente



//fechando o arquivo

fclose(pont\_arq);

**return** 0;

}

**Leitura de arquivos**

**Leitura caracter por caracter - Função fgetc()**

A função **fgetc()** faz a **leitura de um caracter no arquivo**.

Sintaxe: **fgetc(ponteiro\_do\_arquivo);**

Para realizar a **leitura de um arquivo inteiro caracter por caracter** podemos usar **fgetc** dentro de um laço de repetição.

do

{

//faz a leitura do caracter no arquivo apontado por pont\_arq

c = fgetc(pont\_arq);

//exibe o caracter lido na tela

printf("%c" , c);

}while (c != EOF);

A leitura será realizada até que o final do arquivo seja encontrado.

Texto

Descrição gerada automaticamente

Conteúdo do arquivo1.txt

Código do exemplo a seguir.

**#include <stdio.h>**

**#include <stdlib.h>**

**int** main(**void**)

{

FILE \*pont\_arq;

**char** c;

pont\_arq = fopen("arquivo1.txt","r");

**if** (pont\_arq == **NULL**)

{

printf("Erro ao tentar abrir o arquivo!");

exit(1);

}

printf("Lendo e exibindo os dados do arquivo \n\n");

//Faça

**do**

{

c = fgetc(pont\_arq); //faz a leitura do caracter no arquivo apontado por **pont\_arq**

printf("%c" , c); //exibe o caracter lido na tela

} **while** (c != EOF);//enquanto não for final de arquivo

fclose(pont\_arq);//fechando o arquivo

system("pause");//pausa na tela, somente para Windows

**return**(0);

}

**Leitura de strings – Função fgets()**

É **utilizada para leitura de strings em um arquivo**. Realiza a **leitura dos caracteres até o final da linha quando encontra o caracter \n**. A **leitura é efetuada** de tal forma que a **string lida é armazenada em um ponteiro do tipo char**. A função pode ser finalizada **quando encontrar o final do arquivo**, neste caso retorna o endereço da **string lida**. Se ocorrer algum **erro na leitura do arquivo**, a função **retorna NULL**.

//Leitura de arquivo

**#include <stdio.h>**

**#include <stdlib.h>**

**int** main(**void**)

{

FILE \*pont\_arq;

**char** texto\_str[20];

//abrindo o arquivo\_frase em modo "somente leitura"

pont\_arq = fopen("arquivo\_palavra.txt", "r");

//enquanto não for fim de arquivo o looping será executado

//e será impresso o texto

**while**(fgets(texto\_str, 20, pont\_arq) != **NULL**)

printf("%s", texto\_str);

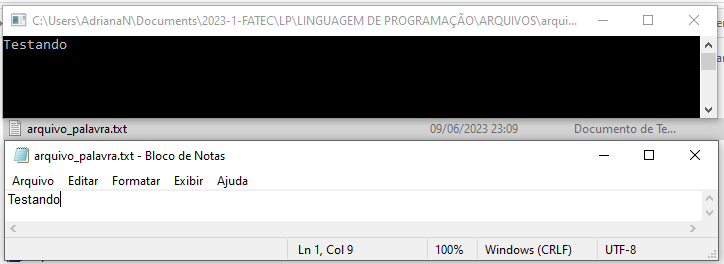
//fechando o arquivo

fclose(pont\_arq);

getch();

**return**(0);

}

****

**Exemplo código que lê de um arquivo que contém uma lista de nomes e escreve as saudações em um outro arquivo**:

#include <stdio.h>

#include <stdlib.h>

int main()

{

// Cria os ponteiros de arquivos.

FILE \*nomes = fopen("names.txt", "r");

FILE \*saudar = fopen("greet.txt", "w");

// Conferir se está tudo certo.

if (!nomes || !saudar)

{

fprintf(stderr, "Erro ao abrir arquivo!\n");

return EXIT\_FAILURE;

}

// Hora das saudações!

char nome[20];

// Basicamente, siga lendo até não haver mais nada para ler.

while (fscanf(nomes, "%s\n", nome) > 0) {

fprintf(saudar, "Olá, %s!\n", nome);

}

if (feof(nomes))

printf("Saudações concluídas!\n");

// Ao chegar ao final, imprima uma mensagem no terminal para informar ao usuário.

fclose(saudar);

fclose(nomes);

saudar = fopen("greet.txt", "r");

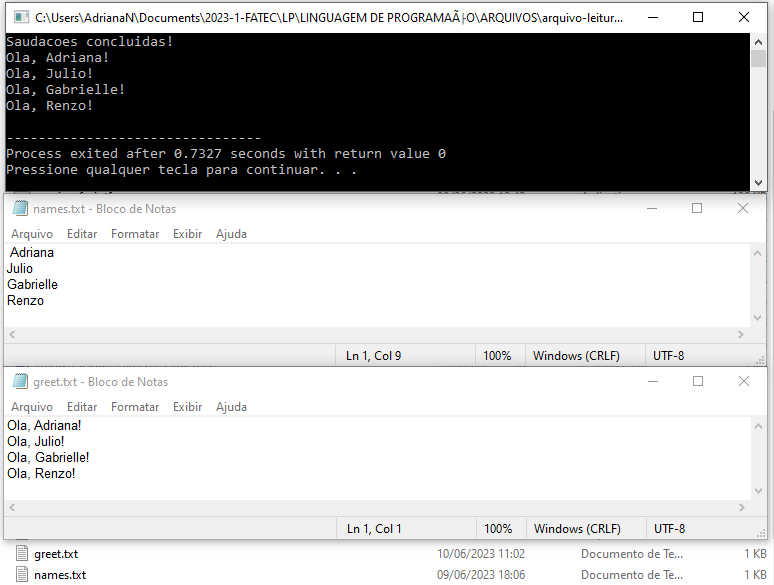
while(fgets(nome, 20, saudar) != NULL)

printf("%s", nome);

fclose(saudar); //fechando o arquivo

return EXIT\_SUCCESS;

}



Exemplo de leitura de dados de um arquivo e como escrever nele:

#include<stdio.h>

#include<conio.h>

main()

{

FILE \*fp;

char ch;

fp = fopen("hello.txt", "w");

printf("Insira dados");

while( (ch = getchar()) != EOF)

{

putc(ch,fp);

}

fclose(fp);

fp = fopen("hello.txt", "r");

while( (ch = getc(fp)!= EOF)

printf("%c",ch);

fclose(fp);

}

A diferença entre getc () e getchar () é que **getc () pode ler de qualquer fluxo de entrada, mas getchar () lê da entrada padrão**. Portanto, getchar () é equivalente a getc (stdin).

Resumindo:

A manipulação de arquivos é uma das partes mais importantes na programação. Em C, usamos um ponteiro de estrutura de um tipo de arquivo para declarar um arquivo:

FILE \*fp

C fornece várias funções integradas para realizar operações básicas de arquivos:

**fopen()**  - criar um arquivo ou abrir um arquivo existente

**fclose()**  - fechar um arquivo

**getc()**  - ler um caractere de um arquivo

**putc()**  - escrever um caractere de um arquivo

**fscanf()**  - ler um conjunto de dados de um arquivo

**fprintf()**  - escrever um conjunto de dados em um arquivo

**getw()**  - ler um número inteiro de um arquivo

**putw()**  - escrever um número inteiro em um arquivo

**fseek()**  - definir a posição em um ponto desejado

**ftell()**  - dar a posição atual no arquivo

**rewind()** - definir a posição como o ponto inicial.

Existem vários modos de abertura de um arquivo:

**r**  - abre um arquivo no modo leitura

**w**  - abre ou cria um arquivo de texto no modo de escrita

**a**  - abre um arquivo no modo de inclusão (append)

**r+** - abre um arquivo nos modos de leitura e escrita

**a+**  - abre um arquivo nos modos de leitura e escrita

**w+** - abre um arquivo nos modos de leitura e escrita

**Arquivos binários em Linguagem C**

**Arquivos binários** são interessantes quando é necessário tratar grande volume de dados, com modificações frequentes, nesse tipo de aplicação eles são mais velozes do que os arquivos do tipo texto e ocupam menos espaço em disco.

Tipos de abertura de arquivos binários

|  |  |
| --- | --- |
| Tipo de abertura | Significado |
| rb | Abre um arquivo binário que já existe**para leitura**. Caso o arquivo não exista, retorna um erro. |
| w+b | Abre um novo arquivo binário **para gravação**. Caso o arquivo já exista será sobrescrito na gravação. |
| a+b | Abre um arquivo binário para adicionar dados. |
| r+b | Abre um arquivo binário que já existe para **leitura e gravação**. Caso o arquivo não exista, retorna um erro. |
| w+b | Abre um novo arquivo binário para **leitura e gravação**. Caso o arquivo já exista será sobrescrito na gravação. |
| a+b | Abre um arquivo binário para adicionar dados. Caso o arquivo não exista, será criado um novo arquivo. |

**Usando frwite() em C**

Como criar um arquivo binário e armazenar dados dentro dele usando a função **fwrite()** da linguagem C.

/\* Testando a função fwrite - Grava um arquivo binário contendo 2 vetores de 15 elementos cada um: O primeiro vetor armazena caracteres e o segundo armazena dados inteiros \*/

**#include <stdio.h>**

**#include <stdlib.h>**

**int** main(**void**)

{

**char** ch [15] = "Teste geral";

**int** valor [15] = {1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15};

FILE \*ptarq;

ptarq = fopen ("binario.bin", "wb"); //Se não abriu, cria.

**if** (ptarq!=**NULL**)

printf ("n\nArquivo binario.bin foi aberto com sucesso\n");

**else**

{

printf ("\n\nErro: O arquivo binario.bin não foi aberto\n");

system ("pause");

exit (0);

}

//Gravando os dados no arquivo usando a função fwrite

fwrite (ch, sizeof (**char**), 15, ptarq);

fwrite (valor, sizeof(**int**), 15, ptarq);

//Fechando o arquivo

fclose (ptarq);

system ("pause");//pausa na tela, somente para Windows

**return**(0);

}

**Leitura de arquivos binários em C**

Agora vamos usar a função **fread()** para fazer a leitura de dados em arquivo binário.

/\* Testando a função fread - Lê um arquivo binário contendo

2 vetores de 15 elementos cada um: O primeiro vetor é de

caracteres e o segundo é de variáveis inteiras \*/

**#include <stdio.h>**

**#include <stdlib.h>**

**int** main(**void**)

{

**char** ch [15];

**int** valor [15], i=0;

FILE \*ptrArq; //ponteiro para o arquivo

ptrArq = fopen("binario.bin","rb"); //abertura do arquivo

**if** (ptrArq != **NULL**) //testando se o arquivo foi aberto com sucesso

printf ("\n\nArquivo binario.bin foi aberto com sucesso\n\n");

**else**

{

printf ("\n\nERRO: O arquivo binario.bin não foi aberto e criado\n");

system ("pause");

exit (1);

}

//leitura do arquivo binário

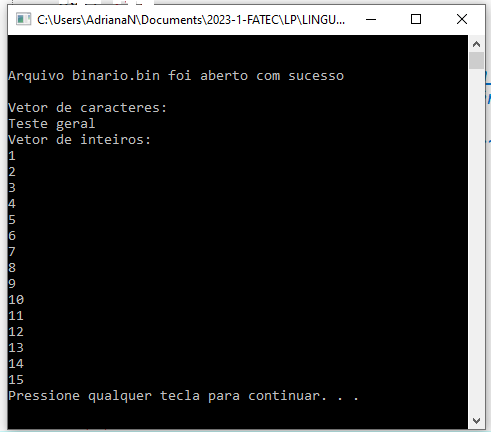
//Sintaxe: fread(&variavel, num\_bytes, num\_registros, arquivo);

//retorna o conteúdo contido em uma ocorrência do tamanho da variável ch.

fread(ch, sizeof(ch),1,ptrArq); //retorna conteúdo contido em uma ocorrência tamanho da variável valor

fread(valor,sizeof(valor),1,ptrArq);

Interface gráfica do usuário, Aplicativo

Descrição gerada automaticamente

printf("Vetor de caracteres: \n");

**for** (i = 0; i < 15; i++)

printf("%c",ch[i]);

printf("\nVetor de inteiros: ");

**for** (i = 0; i < 15; i++)

printf("\n%d",valor[i]);

fclose(ptrArq); //fechando o arquivo

printf("\n");

system ("pause");

**return**(0);

}