# Linguagem C

## Armazenamento de Dados em Arquivos

# Arquivos

- Comandos para gravação e leitura de arquivos:
  - `FILE *arquivo;`
  - `arquivo = fopen(“nome”, “modo”);`
  - `fwrite (&variavel, sizeof (tipo_var), 1, arquivo);`
  - `rewind (arquivo);`
  - `fread (&variavel, sizeof ( tipo_var), 1, arquivo);`
  - `fclose (arquivo);`
  - `feof(arquivo);`

# Arquivos

- Comandos para abertura de arquivos:
  - Definição de variáveis tipo “Arquivo”:

```
FILE *arquivo;
```

  - Abertura (e fechamento) de arquivos:

```
arquivo = fopen(“nome”, “modo”);
if(arquivo!=0) fclose(arquivo);
```

Onde:

nome = nome do arquivo

modo = tipo do arquivo (ascii ou binário), e objetivo de uso (leitura, escrita, anexação)

Obs: o comando fopen retorna um número inteiro: um número maior que zero significa que a abertura foi feita corretamente, 0(zero) indica erro de abertura do arquivo;

Nunca tente fechar um arquivo que não foi aberto!!!

# Arquivos

| Modos de utilização de arquivos | |
|---|---|
| **Modo** | **Significado** |
| r | Abre um arquivo texto para leitura. |
| w | Cria um arquivo texto para escrita. |
| a | Anexa a um arquivo-texto. |
| rb | Abre um arquivo binário para leitura. |
| wb | Cria um arquivo binário para escrita. |
| ab | Anexa um arquivo binário. |
| r+ | Abre um arquivo-texto para leitura/escrita. |
| w+ | Cria um arquivo-texto para leitura/escrita. |
| a+ | Anexa ou cria um arquivo-texto para leitura/escrita. |
| r+b | Abre um arquivo binário para leitura/escrita. |
| w+b | Cria um arquivo binário para leitura/escrita. |
| a+b | Anexa a um arquivo binário para leitura/escrita. |

# Arquivos

- Comando para leitura de dados de arquivos:
  - `fread (&variavel, sizeof ( tipo_var), t, arquivo);`

Onde:

-variavel : variável a ser lida do arquivo (tipos básicos ou compostos);

-tipo_var: o tipo da variável a ser lida do arquivo;

-t é a quantidade de dados a ser lida(1 para uma só variável, mais para leitura de vetores);

-arquivo: variável de arquivo

Obs: o comando fread retorna um número inteiro: um número maior que zero significa que a leitura foi feita corretamente, 0(zero) indica erro de leitura do arquivo;

# Arquivos

- Comando para escrita de dados em arquivos:
  - `fwrite (&variavel, sizeof ( tipo_var), t, arquivo);`

Onde:

-variavel : variável a ser escrita no arquivo (tipos básicos ou compostos);

-tipo_var: o tipo da variável a ser escrita no arquivo;

-t é a quantidade de dados a ser escrita(1 para uma só variável, mais para leitura de vetores);

-arquivo: variável de arquivo

Obs: o comando fwrite retorna um número inteiro: um número maior que zero significa que a escrita foi feita corretamente, 0(zero) indica erro de escrita no arquivo;

# Arquivos

- Comando para re-abertura de arquivos:
  - `rewind(arquivo);`

  Reinicia o ponto de leitura/escrita do arquivo. Ao abrir o arquivo, a cada leitura/escrita, o arquivo vai para a “próxima” variável. O comando rewind reinicia a leitura/escrita a partir da primeira posição do arquivo.

- Comando para verificar o final do arquivo:
  - `feof(arquivo);`

  Obs: o comando feof retorna um número inteiro: um número maior que zero significa que o arquivo está no final, e não há mais dados no arquivo, 0(zero) indica que ainda existem dados;

# Exemplo

Lista Postal

```
/* Um programa de lista postal muito simples */

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

#define TAM 2

struct Elemento
{
       char nome [100];
       char rua [100];
       char cidade [100];
       char estado [2];
       char cep [10];
} Lista [TAM];

char menu ();
void inicia_lista ();
void cadastra ();
void mostra ();
void salva ();
void carrega ();
```

```
int main()
{
char escolha;
 inicia_lista();
 for ( ;; )
 {escolha = menu();
  switch (escolha)
  {case 'c':
   case 'C': { cadastra(); } break;
   case 'm':
   case 'M': { mostra(); } break;
   case 's':
   case 'S': { salva(); } break;
   case 'a':
   case 'A': { carrega(); } break;
   case 't':
   case 'T': { exit (0 ); } break;
   default : { printf ( "Opcao invalida. \n" ); }
  }
 printf ( "\n \n \n" );
}
system ( "Pause" );
return 0;
}
```

```
char menu()
{
char opcao;

 printf ("\n \n \n");
 printf ( " (C)adastrar. \n" );
 printf ( " (M)ostrar. \n" );
 printf ( " C(A)rregar. \n" );
 printf ( " (S)alvar. \n" );
 printf ( " (T)erminar. \n" );
 fflush(stdin);
 scanf ( "%c", &opcao );
 return opcao;
}
```

```
void mostra()
{
int t;
  printf ("\n \n \n");
  for( t = 0; t < TAM; t++ )
  {
  if (!(strcmp(Lista[t].nome, "")==0) )
  {
   printf ( "%s \n", Lista[t].nome);
   printf ( "%s \n", Lista[t].rua);
   printf ( "%s \n", Lista[t].cidade);
   printf ( "%s \n", Lista[t].estado);
   printf ( "%s \n", Lista[t].cep);
  }
  printf ("\n");
  }
}
```

```
void inicia_lista()
{
int t;
for (t = 0; t < TAM; t++)
 {
 strcpy(Lista[t].nome , "");
 }
}
```

```
void cadastra ()
{
int i;
 printf ("\n \n \n");
 for (i = 0; i < TAM; i++ )
 {
    printf ( "Nome: \n" );fflush (stdin);
    gets ( Lista[i].nome );
    printf ( " Rua: \n" );fflush (stdin);
    gets ( Lista[i].rua );
    printf ( "Cidade: \n" );fflush(stdin);
    gets ( Lista[i].cidade );
    printf ( "Estado: \n" );fflush(stdin);
    gets ( Lista[i].estado );
    printf ( "CEP: \n" ); fflush (stdin);
    gets ( Lista[i].cep );
  }
}
```

```
void salva ()
{
FILE *fp;
int i, result;
  printf ("\n \n \n");
  fp = fopen ("cadastro", "wb");
  if ( fp == NULL )
  {
     printf ( "O arquivo nao pode ser aberto. \n" );
     return;
  }
  for (i = 0; i < TAM; i++ )
  {
      if ( !(strcmp(Lista[i].nome, "")==0) )
      {
       result = fwrite ( &Lista[i], sizeof ( struct Elemento ), 1, fp );
       if ( result != 1 )
       {
          printf ( "Erro de escrita no arquivo. \n" );
       }
      }
  }
  fclose (fp);
}
```

```
void carrega ()
{
FILE *fp;
int i, resp;
    printf ("\n \n \n");
    fp = fopen ( "cadastro", "rb" );
    if ( fp == NULL )
    {
       printf ( "O arquivo nao pode ser aberto. \n" );
       return;
    }
    inicia_lista ();
    for (i = 0; i < TAM; i++ )
    {
        resp = fread ( &Lista[i], sizeof (struct Elemento), 1, fp );
        if ( resp != 1 )
        {
         if ( feof (fp) )
         {
            break;
         }
        printf ( " Erro de leitura no arquivo. \n" );
        }
    }
    fclose ( fp );
}
```

- Detalhes:
  - NULL equivale a 0(zero)