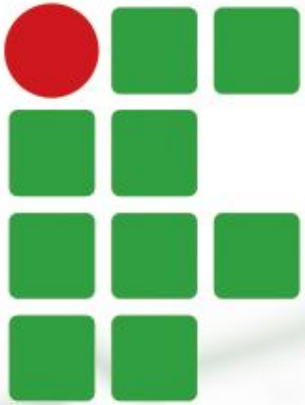

INSTITUTO FEDERAL
Norte de Minas Gerais
Campus Januária

Estruturas de Dados I

- Arquivos de Dados -

Prof. MSc. Adriano Antunes Prates

Persistência de Dados

- Até agora, todos os programas que nós criamos não implementam **persistência de dados**.
  - Ou seja, todos os dados alimentados nos programas **são perdidos assim que ele é encerrado**.
- Uma das formas de implementar persistência é usar **arquivos de dados** para armazenar as informações que são lançadas/geradas pelo sistema.
- A persistência acontece porque os arquivos são **armazenados em memória secundária (não-volátil)**

Utilização de Arquivos

- A utilização de arquivos em *softwares* pode ser útil em três situações...
  - Usar arquivos como **fonte de dados** para a execução do programa (*input*).
  - Usar arquivos como **destino não-volátil** dos dados processados (*output*).
  - Ambas operações (*input* e *output*).

Operações Básicas

- Associação de uma variável a um determinado arquivo físico (em disco) e abertura deste.
- Operações de I/O:
  - Posicionamento
  - Leitura
  - Escrita.
- Fechamento do Arquivo.

Principais Comandos

```
FILE* fopen(char* nomeArquivo, char* modo);
```

- A função **fopen** é responsável pela **abertura de um arquivo físico** em disco. Pode-se utilizar um caminho relativo ou caminho absoluto.

Caminho Relativo...

```
FILE *arq = NULL;
arq = fopen("meu_arquivo.dat", modo);
```

Caminho Absoluto...

```
FILE *arq = NULL;
arq = fopen("/home/adriano/programa/meu_arquivo.dat", modo);
```

Principais Comandos

```
FILE* fopen(char* nomeArquivo, char* modo);
```

- O retorno da função **fopen** é um ponteiro para o tipo ***FILE.***
  - Para o seu programa, um ponteiro FILE é uma representação lógica do arquivo físico em disco.

```
FILE *arq = fopen("meu_arquivo.dat", modo);
```

Principais Comandos

```
FILE* fopen(char* nomeArquivo, char* modo);
```

- O retorno da função **fopen** é um ponteiro para o tipo ***FILE.***
  - Para o seu programa, um ponteiro FILE é uma representação lógica do arquivo físico em disco.

```
FILE *arq = fopen("meu_arquivo.dat", modo);
```

- Outro parâmetro importante é o **MODO** de abertura do arquivo…

Modo Texto e Modo Binário

- No **modo texto**, os dados serão gravados em texto aberto, perceptíveis à leitura humana (como na saída do terminal). Ocupa mais espaço em disco e necessita de caracteres específicos para delimitar os dados (p.ex. arquivos em formato CSV);
- No **modo binário**, os dados serão armazenados analogamente à memória RAM. Ocupa menos espaço em disco e os dados são delimitados pelo tamanho de cada tipo.
- **Usando structs iremos optar por arquivos binários!**

Modo Texto e Modo Binário

- No **modo texto**, os dados serão gravados em texto aberto, percep do termin ssita de caracte (p.ex. arquiv

```
char nome[20] = "Ricardo";
int i = 30;
float a = 1.74;
```

- No **mo** s analog espaço em disco e os dados são delimitados pelo tamanho de cada tipo.

- **Usando structs iremos optar por arquivos binários!**

Modos de Abertura

Modo	Arquivo	Função
"r"	Texto	Leitura. Arquivo deve existir.
"w"	Texto	Escrita. Cria arquivo se não houver. Apaga o anterior se ele existir.
"a"	Texto	Escrita. Os dados serão adicionados no fim do arquivo ("append").
"rb"	Binário	Leitura. Arquivo deve existir.
"wb"	Binário	Escrita. Cria arquivo se não houver. Apaga o anterior se ele existir.
"ab"	Binário	Escrita. Os dados serão adicionados no fim do arquivo ("append").
"r+"	Texto	Leitura/Escrita. O arquivo deve existir e pode ser modificado.
"w+"	Texto	Leitura/Escrita. Cria arquivo se não houver. Apaga o anterior se ele existir.
"a+"	Texto	Leitura/Escrita. Os dados serão adicionados no fim do arquivo ("append").
"r+b"	Binário	Leitura/Escrita. O arquivo deve existir e pode ser modificado.
"w+b"	Binário	Leitura/Escrita. Cria arquivo se não houver. Apaga o anterior se ele existir.
"a+b"	Binário	Leitura/Escrita. Os dados serão adicionados no fim do arquivo ("append").

Problemas...

Modo “ab+” não permite edições

Abrir arquivo em modo “rb+”

Se o arquivo físico não existe!

Abrir arquivo em modo “wb+”

Se o arquivo físico existe!

Exemplo: Abertura de Arquivo

```
#include<stdio.h>

int main(){
    FILE *arq = NULL;
    arq = fopen("testeBSI.txt","rb+");
    if (!arq)
      arq = fopen("testeBSI.txt","wb+");
    if (!arq){
      printf("Programa não conseguiu abrir o arquivo!\n");
      return 0;
    }
    int i = fclose(arq);
    if (!i)
      printf("Arquivo Fechado!");
}
```

Principais Comandos

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

- A função **`fclose`** realiza o fechamento correto e seguro do arquivo apontado pelo ponteiro a.

- Em caso de sucesso, a função retorna o valor **zero**.

Exemplo: Abertura de Arquivo

```
#include<stdio.h>

int main(){
    FILE *arq = NULL;
    arq = fopen("testeBSI.txt","rb+");
    if (!arq)
      arq = fopen("testeBSI.txt","wb+");
    if (!arq){
      printf("Programa não conseguiu abrir o arquivo!\n");
      return 0;
    }
    int i = fclose(arq);
    if (!i)
      printf("Arquivo Fechado!");
}
```

INSTITUTO FEDERAL
Norte de Minas Gerais
Campus Januária

Exemplo: Abertura de Arquivo

```
#include<stdio.h>

int main(){
    FILE *arq = NULL;
    arq = fopen("
                  lose(arq);
    if (!i)
      printf("Arquivo Fechado!");
}
```

Verifique no disco se o arquivo
testeBSI.txt
foi criado com sucesso...

Principais Comandos

```
int fwrite(void* ptr, int size, int n, FILE* arq);
```

- Realiza a escrita de uma estrutura de dados em um arquivo aberto em **modo binário**.

  `ptr` : Ponteiro da estrutura que irá ser escrita.

  `size`: Tamanho em *bytes* da estrutura.

  `n`: A quantidade de cópias que será escrita.

  `arq`: O ponteiro que representa o arquivo destino.

- O retorno da função indica a quantidade de estruturas que foram corretamente escritas.

Exemplo: Escrita em Arquivo

```
void setCarro(FILE* arq){
  Carro novo;
  printf("Digite o Modelo: ");
  scanf(" %[^\n]s", novo.modelo);
  printf("Digite o Ano: ");
  scanf(" %d",&novo.ano);
  if(fwrite(&novo,sizeof(Carro),1,arq))
     printf("Carro Salvo com Sucesso!");
}
```

Principais Comandos

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

- Descarrega o *buffer* em disco. Ou seja, garante que todas as operações realizadas sejam enviadas imediatamente para o disco.

  `arq`: O ponteiro que representa o arquivo destino.

- O retorno da função (0) indica que o flush foi realizado com sucesso.

Recomendado após cada operação fwrite()

Exemplo: Escrita em Arquivo

```
void setCarro(FILE* arq){
  Carro novo;
  printf("Digite o Modelo: ");
  scanf(" %[^\n]s", novo.modelo);
  printf("Digite o Ano: ");
  scanf(" %d",&novo.ano);
  if(fwrite(&novo,sizeof(Carro),1,arq)){
     fflush(arq);
     printf("Carro Salvo com Sucesso!");
  }
}
```

Principais Comandos

```
int fread(void* ptr, int size, int n, FILE* arq);
```

- Realiza a cópia de uma estrutura em arquivo (**modo binário**) para uma estrutura em memória principal.

  `ptr` : Ponteiro para a estrutura que será recuperada.

  `size`: Tamanho em *bytes* da estrutura.

  `n`: A qtde. de elementos que serão recuperados.

  `arq`: O ponteiro que representa o arquivo origem.

- O retorno da função indica a quantidade de estruturas que foram corretamente lidas.

Exemplo: Leitura de Arquivo

```
void getCarro(FILE* arq){
  Carro aux;

  if(fread(&aux,sizeof(Carro),1,arq))
     printf("Modelo: %s\nAno: %d",
                aux.modelo, aux.ano);
}
```

Bora CODAR!!!

- Baseado nos exemplos mostrados, faça um programa que realize uma chamada à função **setCarro()**, e logo após, uma chamada à função **getCarro().**
  - O cadastro foi realizado?
  - A impressão foi realizada?
  - Abra o arquivo físico, e verifique se o registro foi salvo em disco.
  - Execute novamente o programa, mudando os valores de entrada. Abra novamente o arquivo físico. O que aconteceu?

Principais Comandos

```
int fseek(FILE* arq, long salto, int origem);
```

- A função **fseek** efetua o **reposicionamento do cursor do arquivo** indicado pelo ponteiro *arq*.
- Origem:
  - **SEEK_SET** Posiciona cursor no início do arquivo.
  - **SEEK_CUR** Mantém posição atual do cursor.
  - **SEEK_END** Posiciona cursor no fim do arquivo
- Salto:
  - Quantos *bytes* serão deslocados a partir da origem.

Correção da Função setCarro()

```
void setCarro(FILE* arq){
  Carro novo;
  printf("Digite o Modelo: ");
  scanf(" %[^\n]s",novo.modelo);
  printf("Digite o Ano: ");
  scanf(" %d",&novo.ano);
  fseek(arq,0,SEEK_END);
  if(fwrite(&novo,sizeof(Carro),1,arq))
     printf("Carro Salvo com Sucesso!\n");
}
```

Correção da Função getCarro()

```
void getCarro(FILE* arq){
  Carro aux;
  fseek(arq,0,SEEK_SET);
  if (fread(&aux,sizeof(Carro),1,arq))
     printf("Modelo: %s\nAno: %d\n\n",
                    aux.modelo, aux.ano);
}
```

Função getCarros()

- Acessando todos os carros na base de dados...

```
void getCarros(FILE* arq){
  Carro aux;
  fseek(arq,0,SEEK_SET);
  while(fread(&aux,sizeof(Carro),1,arq))
  printf("Modelo: %s\nAno: %d\n\n",
                  aux.modelo, aux.ano);
}
```

Principais Comandos

```
int remove(char* nome_arq);
```

- **`remove()`** permite apagar um arquivo físico (nome_arq) do disco.

- Essa função retorna o valor 0 caso o arquivo seja excluído com sucesso.

INSTITUTO FEDERAL
Norte de Minas Gerais
Campus Januária

Bora CODAR!!!

- Utilizando persistência em arquivos, desenvolva um programa que faça o controle de despesas domésticas.
- Uma despesa possui as seguintes informações: Descrição, Valor, Data de Vencimento e se já está quitada ou ainda não.
- O programa deve oferecer ao usuário as seguintes opções...
  - **Cadastrar Despesa**
  - **Listar todas as Despesas**
    - Quando listar todas as despesas, indicar se ela é a primeira, segunda, terceira, quarta e assim por diante...
  - **Quitar uma Despesa**
    - Nesta opção, o sistema deverá solicitar ao usuário se a despesa a ser quitada é a primeira, segunda, terceira, quarta, etc...
    - Perceba que será necessário ***editar*** um registro no arquivo... Pesquise como fazer isso!
  - **Informar saldo de despesas ainda a pagar.**