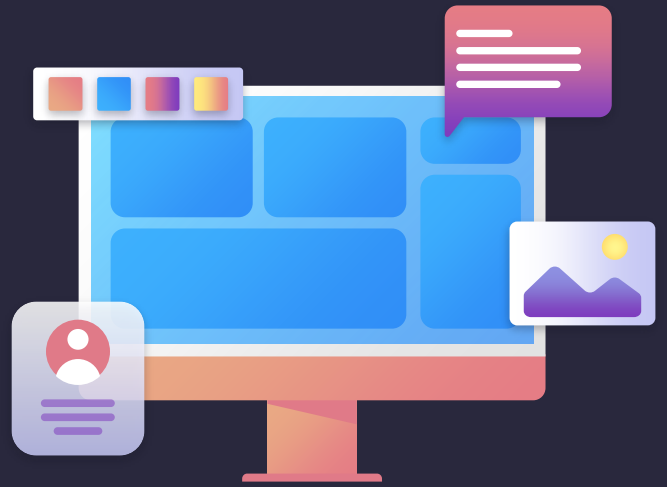
# /Gerenciando Registros em Arquivos Binários

Aimee S. Santos Ferreira
Davi Ventura C. Perdigão
Edmilson Lino Cordeiro
Eric H. de Castro Chaves



# /Arquivos em C

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

2. 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 independente de quais sejam os dados armazenados.

3. Vantagens de utilizar arquivos:
    - **Armazenamento permanente de dados:** as informações permanecem disponíveis mesmo que o programa que as gravou tenha sido encerrado, podendo 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 (geralmente maior do que a RAM).
    - **Acesso concorrente:** Programas podem acessar um arquivo de forma concorrente.

# /TIPOS DE ARQUIVOS

## /01 /TEXTO

Armazena caracteres que podem ser mostrados diretamente na tela ou modificados por um editor de texto **(documentos de texto, código fonte C, páginas XHTML).**

## /02 /BINÁRIO

Sequência de bits que obedece regras do programa que o gerou **(executáveis, arquivos compactados).**

# REGISTROS.C

sistema que gerencia registros que devem ser armazenados em um arquivo binário

# /CRIANDO A ESTRUTURA + FUNÇÕES + MAIN

```c
// O trabalho deverá ser apresentado no dia 13 e er

// Desenvolver em java ou C um sistema que gerencie

// O grupo poderá definir o tema do trabalho, ou se

// Cada registro poderá ser de tamanhos fixos ou va

// O sistema deverá pelo menos: Inserir, listar, bu

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

// Criando a estrutura que será utilizada
You, 2 hours ago | 2 authors (WarLore and others)
typedef struct jogo
{
    char ativo;
    char nome[30];
    char categoria[15];
    int anoLancamento;
    float preco;
} Jogo;

// Protótipo das funções
void escreverArquivo(char nomeArquivo[]);
void listar(char nomeArquivo[]);
void buscar(char nomeArquivo[], char nomeJogo[]);
void remover(char nomeArquivo[], char nomeJogo[]);
```

```c
int main(void)
{
    setlocale(LC_ALL, "Portuguese");
    int opcao;
    char nomeArquivo[] = {"arquivo.bin"};

    do
    {
        printf("\nInforme uma opção:\n");
        printf("\n0 - Sair\n");
        printf("1 - Adicionar jogo\n");
        printf("2 - Listar jogos\n");
        printf("3 - Buscar jogo\n");
        printf("4 - Remover jogo\n");
        scanf("%d", &opcao);
        fflush(stdin);

        switch (opcao)
        {
        case 0:
            printf("\nFinalizando...\n");
            exit(0);
        case 1:
            escreverArquivo(nomeArquivo);
            break;
        case 2:
            listar(nomeArquivo);
            break;
        case 3:
            char nome[30];
            printf("\nInforme o nome:");
            gets(nome);
            buscar(nomeArquivo, nome);
            break;
        case 4:
            printf("Informe o nome:");
            gets(nome);
            remover(nomeArquivo, nome);
            break;
        default:
            printf("Por favor, informe uma opção válida!\n\n");
            break;
        }
    } while (opcao != 0);

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

# /INSERIR DADOS NO ARQUIVO

```c
// Utilizado para escrever no arquivo binário o conteúdo da estrutura
void escreverArquivo(char nomeArquivo[])
{
    Jogo jogo;
    FILE *arquivo = fopen(nomeArquivo, "wb");
    int opcao = 0;
    // Recebendo os dados do usuário e armazenando na estrutura
    if (arquivo)
    {
        do
        {
            int ano;
            float preco;
            jogo.ativo = 'S';
            printf("\n\nInforme o nome:\n");
            scanf("%s", jogo.nome);
            fflush(stdin);

            printf("Informe o gênero:\n");
            scanf("%s", jogo.categoria);
            fflush(stdin);

            printf("Informe o ano de lançamento:\n");
            scanf("%d", &ano);

            if (ano <= 2022 && ano >= 1958)
            { // Tennis for Two
                jogo.anoLancamento = ano;
            }
            else
            {
                jogo.anoLancamento = 2022;
            }

```

- Em C, o arquivo é manipulado através de um ponteiro especial para o arquivo, que tem como função "apontar" a localização de um registro. Sintaxe: **FILE < *ponteiro >** (biblioteca stdio.h)
- Para trabalhar com um arquivo, a primeira operação necessária é abrir este arquivo. Sintaxe: **< ponteiro > = fopen("nome do arquivo", "tipo de abertura");**
- Alguns dos modos de **abertura em arquivos binários** são os seguintes:
  **"rb"**(read): leitura;
  **"rb+"**: leitura e escrita;
  **"wb"**(write): cria um arquivo binário para escrita;
  **"ab"**(append): acrescenta dados binários ao final;
- Ao pegar uma string de entrada com espaços, o buffer não é limpo para a próxima entrada e considera a entrada anterior para o mesmo. Para limpar o fluxo/buffer, utilizamos **fflush(stdin)**

# /INSERIR DADOS NO ARQUIVO

```c
            printf("Informe o preço:\n");
            scanf("%f", &preco);

            if (preco >= 0)
            {
                jogo.preco = preco;
            }
            else
            {
                jogo.preco = 0;
            }

            // Escrevendo o conteúdo da struct no arquivo
            fwrite(&jogo, sizeof(Jogo), 1, arquivo);

            printf("Deseja adicionar mais jogos no momento?  [1 - SIM / 0 - NÃO (VOLTAR AO MENU)]\n");
            fflush(stdin);
            scanf("%d", &opcao);
        } while (opcao != 0);
    }
    else
    {
        printf("Erro ao abrir arquivo!");
    }
    // Fechando o arquivo utilizado
    fclose(arquivo);
}
```

- A função **fwrite()** funciona como a sua companheira **fread()**, porém escrevendo no arquivo.

- O ponteiro passado à **função fclose()** determina o arquivo a ser fechado. A função retorna zero no caso de sucesso. Fechar um arquivo faz com que qualquer caracter que tenha permanecido no "buffer" associado ao fluxo de saída seja gravado.

# /LISTANDO OS DADOS DO ARQUIVO

```c
// Listar todos os dados ativos do arquivo
void listar(char nomeArquivo[])
{
    Jogo jogo;
    FILE *arquivo = fopen(nomeArquivo, "rb");

    if (arquivo)
    {
        // Percorrendo o arquivo e armazenando o conteúdo em uma struct até que não haja mais dados
        while (fread(&jogo, sizeof(Jogo), 1, arquivo))
        {
            if (jogo.ativo == 'S')
            {
                printf("\nNome: %s\nCategoria: %s\nAno de lançamento: %d\nPreço: %.2f\n\n", jogo.nome, jogo.categoria, jogo.anoLancamento, jogo.preco);
            }
        }
        fclose(arquivo);
    }
    else
    {
        printf("Erro ao abrir arquivo!");
    }
}
```

- A função **fread()** retorna o número de unidades efetivamente lidas. Este número pode ser menor que count quando o fim do arquivo for encontrado ou ocorrer algum erro.

# /BUSCAR DADOS NO ARQUIVO

```c
// Buscar um elemento do arquivo usando o nome
void buscar(char nomeArquivo[], char nomeJogo[])
{
    Jogo jogo;
    FILE *arquivo = fopen(nomeArquivo, "rb");
    int i = 0;
    if (arquivo)
    {
        // Percorrendo o arquvivo e armazenando o conteúdo em uma struct até que não haja mais dados
        while (fread(&jogo, sizeof(Jogo), 1, arquivo))
        {
            // Comparando o nome do jogo armazenado na struct com o nome fornecido pelo usuário, caso ele esteja ativo
            if (strcmp(jogo.nome, nomeJogo) == 0 && jogo.ativo == 'S')
            {
                i = 1;
                break;
            }
        }
        if (i == 1)
        { // Se o jogo foi encontrado
            printf("\nNome: %s\nCategoria: %s\nAno de lançamento: %d\nPreço: %.2f\n\n", jogo.nome, jogo.categoria, jogo.anoLancamento, jogo.preco);
        }
        else
        { // Se o jogo não foi encontrado
            printf("\nO jogo não foi encontrado!\n\n");
        }
        fclose(arquivo);
    }
    else
    {
        printf("Erro ao abrir arquivo!");
    }
}
```

- A função **strcamp()** devolve um valor inteiro e indica o relacionamento entre string1 e string2. Um valor menor que zero significa que string1 é menor que string2. Um valor zero significa que ambas são iguais.

# /REMOVER DADOS DO ARQUIVO

```c
// Remover um elemento através da flag e utilizando o nome
void remover(char nomeArquivo[], char nomeJogo[])
{
    Jogo jogo;
    FILE *arquivo = fopen(nomeArquivo, "rb+");
    int i = 0;
    bool estado = false;

    if (arquivo)
    {
        while (fread(&jogo, sizeof(Jogo), 1, arquivo))
        {
            if (strcmp(jogo.nome, nomeJogo) == 0 && jogo.ativo == 'S')
            {
                jogo.ativo = 'N';
                // Apontando para o jogo que foi encontrado
                fseek(arquivo, i * sizeof(Jogo), SEEK_SET);
                // Substituindo na posicao a nova informação
                fwrite(&jogo, sizeof(Jogo), 1, arquivo);
                fclose(arquivo);
                estado = true;
                break;
            }
            i++;
        }
    }
    else
    {
        printf("Erro ao abrir arquivo!");
    }

    if (estado)
    {
        printf("\nO jogo foi removido!\n\n");
    }
    else
    {
        printf("\nO jogo não existe!\n\n");
    }
}
```

- A função **fseek()** nos permite realizar operações de leitura e escrita. Referenciando **SEEK_SET**, inicia-se no começo do arquivo

# <OBRIGADO PELA ATENÇÃO>