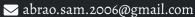
Workshop C - Aula 04

Ponteiros e Structs 04/10/2025

Samuel Abrão

IDP



github.com/samuka7abr

Cronograma Aula 04

- Struct e typedef
- Ponteiros para struct (. e ->)
- Alocação dinâmica (malloc, calloc, realloc, free)
- Manipulação de arquivos (fopen, fclose, fprintf, fscanf)
- Diferença entre arquivos texto e binário
- Exercícios práticos

Struct e typedef

Struct em C

```
// Definindo uma struct
struct Usuario {
    int id;
    char nome[50];
    int idade;
    float salario;
};
// Usando typedef para simplificar
typedef struct Usuario Usuario;
// Declaração de variáveis
Usuario user1;
                                  // Variavel normal
Usuario *user2;
                                  // Ponteiro para struct
Usuario usuarios[10];
                                  // Array de structs
```

Acessando membros da struct

```
// Com variável normal (usar ponto .)
Usuario user;
user.id = 1;
strcpy(user.nome, "João");
user.idade = 25;
user.salario = 3000.50;
printf("ID: %d\n", user.id);
printf("Nome: %s\n", user.nome);
printf("Idade: %d\n", user.idade);
printf("Salário: %.2f\n", user.salario);
```

Operador -> (seta)

Hora da prática!

Questão 1 - Criando lista de usuários

Implemente a função:

Usuario* criarLista(int n);

A função deve alocar dinamicamente memória para um vetor de n usuários.

Retorne o ponteiro para esse vetor.

No main, teste a criação da lista e verifique se a memória foi alocada com sucesso.

Alocação dinâmica

malloc, calloc, realloc, free

```
#include <stdlib.h>
// malloc - aloca memória sem inicializar
int *vetor = malloc(10 * sizeof(int));
// calloc - aloca e inicializa com zero
int *vetor2 = calloc(10, sizeof(int));
// realloc - redimensiona memória já alocada
vetor = realloc(vetor, 20 * sizeof(int));
// free - libera memória alocada
free(vetor):
free(vetor2);
// Para structs
Usuario *lista = malloc(5 * sizeof(Usuario)):
free(lista);
```

Exemplo prático - Lista de usuários

```
//declaração da struct
typedef struct Usuario {
    int id;
    char nome[50];
    int idade;
} Usuario;
//alocação de memória
Usuario *lista = malloc(n * sizeof(Usuario));
//liberação de memória
free(lista);
```

Hora da prática!

Questão 2 - Cadastro de usuário

Implemente a função:

A função deve preencher os dados do usuário na posição indicada.

Use strncpy ou equivalente para copiar o nome.

No main, permita cadastrar manualmente alguns usuários.

Manipulação de arquivos

Abrindo e fechando arquivos

```
#include <stdio.h>
int main() {
    FILE *arquivo;
    // Abrir arquivo para escrita
    arquivo = fopen("dados.txt", "w");
    if (arquivo == NULL) {
        printf("Erro ao abrir arquivo!\n");
        return 1:
    // Escrever no arquivo
    fprintf(arquivo, "Olá, mundo!\n");
    fprintf(arquivo, "Número: %d\n", 42);
    // Fechar arquivo
    fclose(arquivo);
    return 0;
```

Lendo arquivos

```
//abrir arquivo para leitura
arquivo = fopen("dados.txt", "r");
//ler linha por linha
while (fgets(linha, sizeof(linha), arquivo) != NULL) {
    printf("%s", linha);
}
```

Modos de abertura de arquivo

- "r" Leitura (arquivo deve existir)
- "w" Escrita (cria novo arquivo ou sobrescreve)
- "a" Anexar (adiciona ao final do arquivo)
- "r+" Leitura e escrita
- "w+" Escrita e leitura (cria novo arquivo)
- "a+" Anexar e leitura

Importante

Sempre verificar se o arquivo foi aberto com sucesso antes de usar!

Arquivos texto vs binário

Diferenças importantes

Arquivo Texto:

- Legível por humanos
- Usa fprintf/fscanf
- Conversão automática
- Maior tamanho
- Portável entre sistemas

Arquivo Binário:

- Não legível diretamente
- Usa fwrite/fread
- Sem conversão
- Menor tamanho
- Dependente da arquitetura

Hora da prática!

Questão 1 - Criando lista de usuários

Implemente a função:

Usuario* criarLista(int n);

A função deve alocar dinamicamente memória para um vetor de n usuários.

Retorne o ponteiro para esse vetor.

No main, teste a criação da lista e verifique se a memória foi alocada com sucesso.

Questão 2 - Cadastro de usuário

Implemente a função:

A função deve preencher os dados do usuário na posição indicada.

Use strncpy ou equivalente para copiar o nome.

No main, permita cadastrar manualmente alguns usuários.

Questão 3 - Listagem de usuários

Implemente a função:

```
void listarUsuarios(Usuario* lista, int n);
```

A função deve percorrer o vetor de usuários e imprimir id, nome e idade de cada um.

No main, após cadastrar usuários, chame a função para exibir a lista completa.

Questão 4 - Salvando em arquivo texto

Implemente a função:

A função deve abrir o arquivo em modo escrita (w).

Gravar cada usuário em uma linha, no formato:

id;nome;idade

Fechar o arquivo ao final.

Teste a função salvando uma lista de usuários em usuarios.txt.

Questão 5 - Lendo de arquivo texto

Implemente a função:

void lerUsuariosTxt(const char* caminho);

A função deve abrir o arquivo em modo leitura (r).

Ler cada linha com fscanf ou fgets.

Imprimir na tela os dados de cada usuário recuperado do arquivo.

Teste abrindo o arquivo usuarios.txt salvo no exercício anterior.

Obrigado!

Perguntas?