# WESLEY SANTOS DA SILVA BRITO

**WESLEY GABRIEL FERREIRA DE SOUSA**

**MARIA VANUZA DE SOUSA SILVA**

# Lista de Exercícios - Structs e Ponteiros em C

Esta lista de exercícios aborda conceitos fundamentais de **Structs e Ponteiros** em C. Os exercícios podem ser realizados **individualmente, em dupla ou em trio**.

Cada questão vale **0,5 ponto**, totalizando uma pontuação máxima de **7 pontos**.

⚠️ **Critérios para validação da lista:**

* A lista **somente será considerada válida** se o aluno apresentar e explicar sua resolução ao professor (*Obs: o professor sorteará aleatoriamente questões para os alunos explicarem*).
* **Alunos que concluírem a lista, mas não apresentarem, receberão nota 0** nesta atividade.

💡 **Dica:** Certifique-se de compreender bem os conceitos envolvidos, pois a explicação da resolução será parte da avaliação.

## Parte 1 - Structs (7 questões)

### Criando e Manipulando Structs

Crie uma struct `Livro` com os campos `titulo`, `autor` e `ano\_publicacao`. Leia os dados do teclado e imprima-os na tela.

#include <stdio.h>

#include <string.h>

#include <locale.h>

struct Livro {

char titulo[100];

char autor[100];

int ano\_publicacao;

};

int main() {

setlocale(LC\_ALL, "portuguese");

struct Livro livro;

printf("Digite o título do livro: ");

fgets(livro.titulo, sizeof(livro.titulo), stdin);

livro.titulo[strcspn(livro.titulo, "\n")] = '\0';

printf("Digite o autor do livro: ");

fgets(livro.autor, sizeof(livro.autor), stdin);

livro.autor[strcspn(livro.autor, "\n")] = '\0';

printf("Digite o ano de publicação do livro: ");

scanf("%d", &livro.ano\_publicacao);

printf("\nDados do Livro:\n");

printf("Título: %s\n", livro.titulo);

printf("Autor: %s\n", livro.autor);

printf("Ano de Publicação: %d\n", livro.ano\_publicacao);

return 0;

}

### Array de Structs

Crie um vetor de 5 registros da struct `Carro`, contendo `marca`, `modelo` e `ano`. Peça ao usuário para preencher os dados e exiba todos os carros cadastrados.

#include <stdio.h>

#include <string.h>

#include <locale.h>

struct Carro {

char marca[50];

char modelo[50];

int ano;

};

int main() {

setlocale(LC\_ALL, "portuguese");

struct Carro carros[5];

for (int i = 0; i < 5; i++) {

printf("\nCadastro do Carro %d:\n", i + 1);

printf("Digite a marca do carro: ");

fgets(carros[i].marca, sizeof(carros[i].marca), stdin);

carros[i].marca[strcspn(carros[i].marca, "\n")] = '\0';

printf("Digite o modelo do carro: ");

fgets(carros[i].modelo, sizeof(carros[i].modelo), stdin);

carros[i].modelo[strcspn(carros[i].modelo, "\n")] = '\0';

printf("Digite o ano do carro: ");

scanf("%d", &carros[i].ano);

getchar();

}

printf("\nCarros Cadastrados:\n");

for (int i = 0; i < 5; i++) {

printf("\nCarro %d:\n", i + 1);

printf("Marca: %s\n", carros[i].marca);

printf("Modelo: %s\n", carros[i].modelo);

printf("Ano: %d\n", carros[i].ano);

}

return 0;

}

### Structs e Funções

Crie uma struct `Produto` com os atributos `nome`, `preco` e `quantidade`. Escreva uma função `void exibirProduto(Produto p)` que recebe um `Produto` como parâmetro e imprime suas informações. Teste a função chamando-a no `main()`.

#include <stdio.h>

#include <string.h>

#include <locale.h>

struct Produto {

char nome[50];

float preco;

int quantidade;

};

void exibirProduto(struct Produto p) {

setlocale(LC\_ALL, "portuguese");

printf("\nInformações do Produto:\n");

printf("Nome: %s\n", p.nome);

printf("Preço: R$ %.2f\n", p.preco);

printf("Quantidade: %d\n", p.quantidade);

}

int main() {

setlocale(LC\_ALL, "portuguese");

struct Produto produto;

printf("Digite o nome do produto: ");

fgets(produto.nome, sizeof(produto.nome), stdin);

produto.nome[strcspn(produto.nome, "\n")] = '\0';

printf("Digite o preço do produto: ");

scanf("%f", &produto.preco);

printf("Digite a quantidade do produto: ");

scanf("%d", &produto.quantidade);

exibirProduto(produto);

return 0;

}

### Struct dentro de Struct

Defina uma struct `Endereco` com os campos `rua`, `numero` e `cidade`. Defina uma struct `Pessoa` que contenha um `Endereco` e os atributos `nome` e `idade`. Leia os dados de uma pessoa e seu endereço e exiba-os na tela.

#include <stdio.h>

#include <string.h>

#include <locale.h>

struct Endereco {

char rua[100];

int numero;

char cidade[100];

};

struct Pessoa {

char nome[100];

int idade;

struct Endereco endereco;

};

int main() {

setlocale(LC\_ALL, "portuguese");

struct Pessoa pessoa;

printf("Digite o nome da pessoa: ");

fgets(pessoa.nome, sizeof(pessoa.nome), stdin);

pessoa.nome[strcspn(pessoa.nome, "\n")] = '\0';

printf("Digite a idade da pessoa: ");

scanf("%d", &pessoa.idade);

getchar();

printf("\nDigite os dados do endereço:\n");

printf("Rua: ");

fgets(pessoa.endereco.rua, sizeof(pessoa.endereco.rua), stdin);

pessoa.endereco.rua[strcspn(pessoa.endereco.rua, "\n")] = '\0';

printf("Número: ");

scanf("%d", &pessoa.endereco.numero);

getchar();

printf("Cidade: ");

fgets(pessoa.endereco.cidade, sizeof(pessoa.endereco.cidade), stdin);

pessoa.endereco.cidade[strcspn(pessoa.endereco.cidade, "\n")] = '\0';

printf("\nDados da Pessoa:\n");

printf("Nome: %s\n", pessoa.nome);

printf("Idade: %d\n", pessoa.idade);

printf("\nEndereço:\n");

printf("Rua: %s, %d\n", pessoa.endereco.rua, pessoa.endereco.numero);

printf("Cidade: %s\n", pessoa.endereco.cidade);

return 0;

}

### Vetores dentro de Structs

Crie uma struct `Aluno` com os atributos `nome`, `matricula` e um vetor de 3 notas. Leia os dados de um aluno e exiba a média das três notas.

#include<stdio.h>

#include<locale.h>

#include<stdlib.h>

#include<string.h>

struct aluno{

char nome[50];

int matricula;

float notas[3];

};

int main(){

setlocale(LC\_ALL, "Portuguese");

struct aluno aluno;

printf("Digite o nome do aluno: ");

scanf("%s", aluno.nome);

printf("Digite a matricula do aluno: ");

scanf("%d", &aluno.matricula);

for(int i=0; i<3; i++){

printf("Digite a nota %d: ", i+1);

scanf("%f", &aluno.notas[i]);

}

float soma = 0;

for(int i=0; i<3; i++){

soma += aluno.notas[i];

}

float media = soma/3;

printf("Nome: %s\n", aluno.nome);

printf("Matricula: %d\n", aluno.matricula);

printf("Média das notas: %.2f\n", media);

return 0;

}

### Cadastro e Busca em Structs

Crie um vetor de `Produto`, onde cada produto tem um `codigo` e um `preco`. Peça ao usuário para digitar um código e busque o produto correspondente no vetor. Caso o código não exista, exiba uma mensagem de erro.

#include<stdio.h>

#include<stdlib.h>

struct Produto{

int codigo;

float preco;

};

int main()

{

int n, CodigoBusca;

int encontrado=0;

printf("Digite o numero de produtos: ");

scanf("%d", &n);

struct Produto\*produtos = (struct Produto\*)malloc(n\*sizeof(struct Produto));

for(int i=0; i<n; i++){

printf("Digite o codigo do produto %d: ", i+1);

scanf("%d", &produtos[i].codigo);

printf("Digite o preco do produto %d: ", i+1);

scanf("%f", &produtos[i].preco);

}

printf("Digite o codigo do produto que deseja buscar: ");

scanf("%d", &CodigoBusca);

for(int i=0; i<n; i++){

if(produtos[i].codigo == CodigoBusca){

printf("Produto encontrado!\n");

printf("Codigo: %d\n", produtos[i].codigo);

printf("Preco: %.2f\n", produtos[i].preco);

encontrado = 1;

break;

}

}

if(!encontrado){

printf("Erro: Produto com código %d não encontrado.\n", CodigoBusca);

}

free(produtos);

return 0;

}

### Alteração de Dados em Structs

Use a struct `Aluno` e crie uma função `void atualizarNota(Aluno \*a, int indice, float novaNota)` que altera uma das notas do aluno. Teste a função no `main()`.

#include<stdio.h>

#include<stdlib.h>

#include<string.h>

#include<locale.h>

struct Aluno{

char nome[50];

int matricula;

float notas[3];

};

void atualizarNota(struct Aluno \*a, int indice, float novaNota){

if(indice>=0 && indice<3){

a->notas[indice] = novaNota;

} else {

printf("indice invalido. deve ser entre 0 e 2\n");

}

}

int main(){

struct Aluno aluno;

setlocale(LC\_ALL, "Portuguese");

printf("Digite o nome do aluno: ");

scanf(" %[^\n]", &aluno.nome);

printf("Digite a matricula do aluno: ");

scanf("%d", &aluno.matricula);

for(int i=0; i<3; i++){

printf("Digite a nota %d do aluno: ", i+1);

scanf("%f", &aluno.notas[i]);

}

printf("\n Notas antes da atualizacao:\n");

for(int i=0; i<3; i++){

printf("Nota %d: %.2f\n", i+1, aluno.notas[i]);

}

int indice;

float novaNota;

printf("\nDigite o indice da nota que deseja atualizar: ");

scanf("%d", &indice);

printf("Digite a nova nota: ");

scanf("%f", &novaNota);

atualizarNota(&aluno, indice, novaNota);

printf("\n Nota apos a atualizacao:\n");

for(int i=0; i<3; i++){

printf("Nota %d: %.2f\n", i+1, aluno.notas[i]);

}

}

## Parte 2 - Ponteiros (7 questões)

### Ponteiros Simples

Declare um ponteiro para um inteiro, atribua um valor a uma variável inteira e use o ponteiro para imprimir esse valor.

#include<stdio.h>

#include<locale.h>

int main(){

int var= 10;

int \*ptr;

float var2= 12;

float \*ptr2;

ptr = &var;

ptr2 = &var2;

printf("Valor de var: %d\n", var);

printf("valor armazenado no ponteiro ptr2: %2.f\n", var2);

}

### Ponteiros e Funções

Crie uma função `void dobrar(int \*n)` que dobra o valor de um número usando ponteiros. Teste a função no `main()`.

#include<stdio.h>

#include<locale.h>

void dobrar (int \*n){

\*n = \*n \* 2;

}

int main(){

setlocale(LC\_ALL, "Portuguese");

int var= 10;

int \*ptr;

float var2= 12;

float \*ptr2;

ptr = &var;

ptr2 = &var2;

printf("Valor de var: %d\n", var);

printf("valor armazenado no ponteiro var: %d\n", \*ptr);

printf("valor de var antes de dobrar: %d\n", var);

dobrar(&var);

printf("valor de var depois de dobrar: %d\n", var);

}#include<stdio.h>

#include<locale.h>

void dobrar (int \*n){

\*n = \*n \* 2;

}

int main(){

setlocale(LC\_ALL, "Portuguese");

int var= 10;

int \*ptr;

float var2= 12;

float \*ptr2;

ptr = &var;

ptr2 = &var2;

printf("Valor de var: %d\n", var);

printf("valor armazenado no ponteiro var: %d\n", \*ptr);

printf("valor de var antes de dobrar: %d\n", var);

dobrar(&var);

printf("valor de var depois de dobrar: %d\n", var);

}

### Ponteiros e Structs

Declare um ponteiro para uma struct `Pessoa`, aloque memória dinamicamente com `malloc()` e preencha os dados dela. Depois, exiba os dados na tela e libere a memória.

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

#include <locale.h>

struct Pessoa {

char nome[50];

int idade;

float altura;

};

int main() {

setlocale(LC\_ALL, "portuguese");

struct Pessoa \*pessoa;

pessoa = (struct Pessoa \*) malloc(sizeof(struct Pessoa));

if (pessoa == NULL) {

printf("Erro ao alocar memória!\n");

return 1;

}

strcpy(pessoa->nome, "João Silva");

pessoa->idade = 25;

pessoa->altura = 1.75;

printf("Dados da Pessoa:\n\n");

printf("Nome: %s\n", pessoa->nome);

printf("Idade: %d\n", pessoa->idade);

printf("Altura: %.2f\n", pessoa->altura);

free(pessoa);

return 0;

}

### Alocação Dinâmica de Vetores

Solicite ao usuário um tamanho `N` para um vetor de inteiros. Use `malloc()` para alocar dinamicamente o vetor e preencha os valores com entrada do usuário. Exiba os valores e libere a memória alocada.

#include <stdio.h>

#include <stdlib.h>

#include <locale.h>

int main() {

setlocale(LC\_ALL, "portuguese");

int n;

int \*vetor;

vetor = &n;

printf("Digite o valor do vetor: ");

scanf("%d", &n);

printf("\n");

vetor = (int \*) malloc(n \* sizeof(int));

if (vetor == NULL) {

printf("Erro ao alocar memória!\n");

return 1;

}

for (int i = 0; i < n; i++) {

printf("Digite o valor de %d: ", i);

scanf("%d", &vetor[i]);

}

printf("\n");

for (int i = 0; i < n; i++) {

printf("[%d]", vetor[i]);

}

free(vetor);

return 0;

}

### Array de Structs com Ponteiros

Crie um vetor dinâmico de `Aluno` usando `malloc()`. Peça ao usuário o número de alunos e preencha seus dados. Exiba as informações e libere a memória.

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

#include <locale.h>

typedef struct{

char nome[50];

int idade;

float nota;

}aluno;

int main() {

setlocale(LC\_ALL, "portuguese");

int numAlunos;

aluno \*alunos;

printf("Digite o número de alunos: ");

scanf("%d", &numAlunos);

alunos = (aluno \*)malloc(numAlunos \* sizeof(aluno));

if (alunos == NULL) {

printf("Erro ao alocar memória!\n");

return 1;

}

for (int i = 0; i < numAlunos; i++) {

printf("\nDADOS DO %dº ALUNO\n", i + 1);

printf("Nome: ");

scanf("%s", alunos[i].nome);

printf("Idade: ");

scanf("%d", &alunos[i].idade);

printf("Nota: ");

scanf("%f", &alunos[i].nota);

}

printf("\nINFORMAÇÕES DOS ALUNOS\n");

for (int i = 0; i < numAlunos; i++) {

printf("%dº ALUNO\n", i + 1);

printf("Nome: %s\n", alunos[i].nome);

printf("Idade: %d\n", alunos[i].idade);

printf("Nota: %.2f\n", alunos[i].nota);

printf("\n");

}

free(alunos);

return 0;

}

### Modificando Structs com Ponteiros

Crie uma função `void alterarPreco(Produto \*p, float novoPreco)` que recebe um ponteiro para um `Produto` e altera seu preço. No `main()`, teste a função chamando-a para modificar um produto.

#include <stdio.h>

#include <locale.h>

typedef struct {

char nome[50];

float preco;

} Produto;

void alterarPreco(Produto \*p, float novoPreco) {

p->preco = novoPreco;

}

int main() {

setlocale(LC\_ALL, "portuguese");

Produto produto = {"Celular", 2000.00};

printf("Produto: %s\n\n", produto.nome);

printf("Preço original: R$ %.2f\n", produto.preco);

alterarPreco(&produto, 1500.00);

printf("Novo preço: R$ %.2f\n", produto.preco);

return 0;

}

### Questão - Vetor Dinâmico de Alunos com Cálculo de Média

Escreva um programa em C que:

1. Solicite ao usuário a quantidade de alunos a serem cadastrados.
2. Utilize **malloc()** para alocar dinamicamente um vetor de structs Aluno.
3. A struct Aluno deve conter os seguintes campos:

char nome[50]; float nota1; float nota2; float media;

1. Para cada aluno, o programa deve solicitar o **nome**, as **duas notas** e calcular a **média**.
2. Após a entrada dos dados, o programa deve exibir as informações de todos os alunos cadastrados, incluindo suas médias.
3. Por fim, o programa deve **liberar a memória alocada**.

#include <stdio.h>

#include <stdlib.h>

#include <locale.h>

typedef struct {

char nome[50];

float nota1;

float nota2;

float media;

} Aluno;

int main() {

setlocale(LC\_ALL,"portuguese");

int numAlunos;

printf("Digite a quantidade de alunos: ");

scanf("%d", &numAlunos);

Aluno \*alunos = (Aluno \*)malloc(numAlunos \* sizeof(Aluno));

if (alunos == NULL) {

printf("Erro ao alocar memória!\n");

return 1;

}

for (int i = 0; i < numAlunos; i++) {

printf("\nAluno %d:\n", i + 1);

printf("Nome: ");

scanf("%s", alunos[i].nome);

printf("Nota 1: ");

scanf("%f", &alunos[i].nota1);

printf("Nota 2: ");

scanf("%f", &alunos[i].nota2);

alunos[i].media = (alunos[i].nota1 + alunos[i].nota2) / 2;

}

printf("\nDados dos alunos:\n");

for (int i = 0; i < numAlunos; i++) {

printf("\nAluno %d:\n", i + 1);

printf("Nome: %s\n", alunos[i].nome);

printf("Nota 1: %.2f\n", alunos[i].nota1);

printf("Nota 2: %.2f\n", alunos[i].nota2);

printf("Média: %.2f\n", alunos[i].media);

}

free(alunos);

return 0;

}