**Conceitos Iniciais: structs – Exemplo Sintaxe**

#include <stdio.h> // Biblioteca padrão de entrada e saída (printf, scanf etc.)

// Definição da estrutura 'pessoa'

struct pessoa {

int idade; // Campo para armazenar a idade (inteiro)

float altura; // Campo para armazenar a altura (número real)

};

// Criação de um apelido 'Pessoa' para 'struct pessoa'

typedef struct pessoa Pessoa;

int main() {

// Declaração de uma variável do tipo Pessoa

Pessoa p;

// Atribuição de valores aos campos da estrutura

p.idade = 5; // Define a idade como 5

p.altura = 1.65; // Define a altura como 1.65 metros

// Impressão dos valores armazenados na estrutura

printf("A idade da pessoa é: %d", p.idade); // Mostra a idade

printf("\nA altura da pessoa é: %.2f", p.altura); // Mostra a altura com 2 casas decimais

return 0; // Fim do programa

}

**Exemplo 2:**

#include <stdio.h> // Biblioteca padrão usada para entrada e saída (ex: printf, scanf)

// Definindo a estrutura chamada 'Pessoa'

struct Pessoa {

char nome[50]; // Campo para armazenar o nome da pessoa (até 49 caracteres + '\0')

int idade; // Campo para a idade da pessoa (tipo inteiro)

float altura; // Campo para a altura da pessoa (tipo float)

};

int main() {

struct Pessoa pessoas[3]; // Vetor com 3 posições, para armazenar dados de 3 pessoas

int i; // Variável usada para controlar os loops (for)

// Loop para coletar os dados de 3 pessoas

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

printf("\n--- Pessoa %d ---\n", i + 1); // Informa ao usuário qual pessoa está sendo cadastrada

printf("Digite o nome: ");

scanf("%49s", pessoas[i].nome); // Lê o nome e armazena no campo 'nome' da struct

printf("Digite a idade: ");

scanf("%d", &pessoas[i].idade); // Lê a idade e armazena no campo 'idade'

printf("Digite a altura (em metros): ");

scanf("%f", &pessoas[i].altura); // Lê a altura e armazena no campo 'altura'

}

// Exibição dos dados coletados

printf("\n\n===== Lista de Pessoas Cadastradas =====\n");

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

printf("\nPessoa %d:\n", i + 1); // Mostra o número da pessoa na Lista

printf("Nome: %s\n", pessoas[i].nome); // Exibe o nome armazenado

printf("Idade: %d anos\n", pessoas[i].idade); // Exibe a idade armazenada

printf("Altura: %.2f metros\n", pessoas[i].altura); // Exibe a altura com duas casas decimais

}

return 0; // Encerra o programa com sucesso

}

**Exemplo 3:**

#include <stdio.h> // Biblioteca padrão para entrada e saída de dados

// Definindo a estrutura Endereço

struct Endereço {

char rua[50]; // Campo para armazenar o nome da rua

int numero; // Campo para armazenar o número da residência

};

// Definindo a estrutura Pessoa, que contém um campo do tipo Endereço

struct Pessoa {

char nome[50]; // Campo para armazenar o nome da pessoa

int idade; // Campo para armazenar a idade da pessoa

struct Endereço endereço; // Campo do tipo Endereço (struct aninhada)

};

int main() {

struct Pessoa p; // Declarando uma variável do tipo Pessoa

// Solicitando e lendo o nome da pessoa (sem espaços)

printf("Digite o nome: ");

scanf("%49s", p.nome); // Lê uma string (limite de 49 caracteres)

// Solicitando e lendo a idade da pessoa

printf("Digite a idade: ");

scanf("%d", &p.idade); // Lê um número inteiro

// Solicitando e lendo o nome da rua (sem espaços)

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

scanf("%49s", p.endereço.rua); // Lê uma string (limite de 49 caracteres)

// Solicitando e lendo o número da residência

printf("Digite o número da casa: ");

scanf("%d", &p.endereço.numero); // Lê um número inteiro

// Exibindo os dados coletados

printf("\n--- Dados Cadastrados ---\n");

printf("Nome: %s\n", p.nome); // Exibe o nome da pessoa

printf("Idade: %d\n", p.idade); // Exibe a idade da pessoa

printf("Endereço: Rua %s, Nº %d\n", p.endereço.rua, p.endereço.numero); // Exibe o endereço completo

return 0; // Indica que o programa terminou com sucesso

}

**POR QUE USAMOS PONTEIROS?**

#include <stdio.h> // Biblioteca padrão para entrada e saída

// Função que dobra o valor de uma variável usando ponteiros void dobrar(int \*num) { \*num = \*num \* 2; // Acessa o valor apontado e o dobra }

int main() { int valor = 10; // Declara uma variável inteira e atribui o valor 10

printf("Antes da função: %d\n", valor); // Exibe o valor original (10)  
dobrar(&valor); // Passa o endereço da variável para a função  
printf("Depois da função: %d\n", valor); // Exibe o valor modificado (20)  
  
return 0; // Fim do programa

}

**Exemplo 1:**

#include <stdio.h>

#include <string.h>

// Definição da struct Pessoa

struct Pessoa {

char nome[100];

int idade;

float altura;

};

// Função que recebe uma cópia da struct (passagem por valor)

void atualizarIdade(struct Pessoa p) {

p.idade += 1; // Modifica apenas a cópia local, NÃO afeta o original

}

int main() {

struct Pessoa pessoa;

// Inicialização dos dados da pessoa

strcpy(pessoa.nome, "Ana Paula"); // Copia o nome para a struct

pessoa.idade = 25; // Atribui a idade

pessoa.altura = 1.65; // Atribui a altura

printf("Antes (por valor): %d anos\n", pessoa.idade); // Exibe 25

atualizarIdade(pessoa); // Passagem por valor (cópia da struct)

printf("Depois (por valor): %d anos\n", pessoa.idade); // Continua 25 (não alterado)

return 0;

}

**Exemplo 2:**

#include <stdio.h>

#include <string.h>

// Definição da struct Pessoa

struct Pessoa {

char nome[100];

int idade;

float altura;

};

// Função que recebe um ponteiro para struct Pessoa

void atualizarIdade(struct Pessoa \*p) {

p->idade += 1; // Acesso diretamente o campo idade e incrementa

}

int main() {

struct Pessoa pessoa; // Correção: 'Pessoa' em vez de 'pessoa.pessoa'

strcpy(pessoa.nome, "Ana Paula");

pessoa.idade = 25;

pessoa.altura = 1.65;

printf("Antes (por ponteiro): %d anos\n", pessoa.idade);

atualizarIdade(&pessoa); // Passa o ENDEREÇO da struct

printf("Depois (por ponteiro): %d anos\n", pessoa.idade); // Agora a idade foi alterada!

return 0;

}

**Relembrando: Alocação Dinâmica**

#include <stdio.h> // Inclui a biblioteca padrão de entrada e saída (para printf) #include <stdlib.h> // Inclui a biblioteca padrão de alocação de memória (para malloc e free)

// Definindo uma estrutura chamada "ponto" com dois campos do tipo float struct ponto { float x; // Coordenada x do ponto float y; // Coordenada y do ponto };

// Criando um apelido "Ponto" para "struct ponto", facilitando a declaração de variáveis typedef struct ponto Ponto;

int main() { // Aloca dinamicamente memória suficiente para armazenar um struct Ponto // e faz o ponteiro 'p' apontar para essa área de memória Ponto *p = (Ponto*) malloc(sizeof(Ponto));

// Atribui o valor 1.0 ao campo 'x' do struct apontado por 'p'  
p->x = 1.0;  
  
// Atribui o valor 5.0 ao campo 'y' do struct apontado por 'p'  
p->y = 5.0;  
  
// Imprime as coordenadas do ponto com duas casas decimais usando printf  
printf("Ponto = (%.2f, %.2f)\n", p->x, p->y); // Adicionado o parêntese fechando  
  
// Libera a memória que foi alocada dinamicamente para evitar vazamentos de memória  
free(p);  
  
// Encerra o programa com retorno 0 (sucesso)  
return 0;

}