ESTRUTURA DE DADOS

Curso de Licenciatura em Ciências da Computação

Lucas Sampaio Leite

Structs

INSTITUTO FEDERAL
Baiano

- Tipos básicos (primitivos): Armazenam um único valor por variável.
  - char, int, float, double

```
int idade;
float altura;
char sexo;
```

- Tipos compostos homogêneos: Armazenam vários valores do mesmo tipo.
  - Vetores e matrizes (arrays)

```
char nome[50];
char frequencia[12][30];
```

Structs

INSTITUTO FEDERAL
Baiano

- Em C, uma struct (abreviação de structure) é um tipo de dado composto que permite agrupar variáveis de diferentes tipos sob um único nome.
- As estruturas são consideradas tipos de dados derivados, pois são construídas a partir de tipos já existentes na linguagem. Elas permitem definir um novo tipo que reúne vários membros (campos), cada um podendo possuir um tipo de dado diferente.

```c
struct Aluno {
    char nome[50];
    int matricula;
    float nota;
};
```

Structs

INSTITUTO FEDERAL
Baiano

- A linguagem C permite criar nossos próprios tipos de variáveis;
- Ou seja, podemos definir novos tipos de dados de acordo com a necessidade do programa;
- Isso é útil quando queremos representar entidades mais complexas, formadas por vários atributos.
- Sintaxe:

```
struct NomeDaStruct {
    tipo1 membro1;
    tipo2 membro2;

    tipo3 membroN;
};
```

Structs

INSTITUTO FEDERAL
Baiano

- A linguagem C permite criar nossos próprios tipos de variáveis;
- Ou seja, podemos definir novos tipos de dados de acordo com a necessidade do programa;
- Isso é útil quando queremos representar entidades mais complexas, formadas por vários atributos.
- Sintaxe:

```
struct NomeDaStruct {
    tipo1 membro1;
    tipo2 membro2;

    tipo3 membroN;
};
```

O nome da struct é definido na primeira linha, e seus membros são declarados dentro das chaves, nas linhas seguintes.

Structs

INSTITUTO FEDERAL
Baiano

- A linguagem C criar nossos próprios tipos de variáveis;
- Ou seja, podemos definir novos tipos de dados de acordo com a necessidade do programa;
- Isso é útil quando queremos representar entidades mais complexas, formadas por vários atributos.
- Sintaxe:

```
struct NomeDaStruct {
    tipo1 membro1;
    tipo2 membro2;

    tipo3 membroN;
};
```

Cada membro é declarado da mesma forma que uma variável em C: primeiro vem o tipo de dado, seguido do nome do membro.

Structs

```c
#include <stdio.h>

struct Pessoa{
    char nome[50], endereco[50];
    int id, idade;
    char sexo;
};

int main() {
    struct Pessoa p1;

    return 0;
}
```

Ok... criei p1. E agora?

INSTITUTO FEDERAL
Baiano

Structs

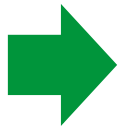
```c
#include <stdio.h>
#include <string.h>

struct Pessoa{
    char nome[50], endereco[50];
    int id, idade;
    char sexo;
};

int main() {
    struct Pessoa p1;

    strcpy(p1.nome, "Lucas");
    strcpy(p1.endereco, "Rua 1");
    p1.id = 1;
    p1.idade = 18;
    p1.sexo = 'M';

    return 0;
}
```

INSTITUTO FEDERAL
Baiano

Structs

INSTITUTO FEDERAL
Baiano

```c
struct Pessoa p1;

strcpy(p1.nome, "Lucas");
strcpy(p1.endereco, "Rua 1");
p1.id = 1;
p1.idade = 18;
p1.sexo = 'M';

printf("nome: %s \n", p1.nome);
printf("idade: %d \n", p1.idade);
```



```
nome: Lucas
idade: 18
```

Structs

- É possível declarar variáveis no mesmo momento em que a estrutura é definida.
- Exemplo:

```
struct NomeDaStruct {
    tipo1 membro1;
    tipo2 membro2;

    tipo3 membroN;
} variavel1, variavel2, variavel3;
```

Após a chave de fechamento da struct, podemos informar os nomes das variáveis que serão criadas com aquele novo tipo.

Structs

```c
struct Pessoa{
    char nome[50], endereco[50];
    int id, idade;
    char sexo;
};

int main() {
    struct Pessoa p1, p2, p3, p4;

    return 0;
}
```

```c
int main() {
    char nomeP1[50], nomeP2[50], nomeP3[50], nomeP4[50];
    char enderecoP1[50], enderecoP2[50], enderecoP3[50], enderecoP4[50];
    int idP1, idP2, idP3, idP4;
    int idadeP1, idadeP2, idadeP3, idadeP4;
    char sexoP1, sexoP2, sexoP3, sexoP4;

    return 0;
}
```

INSTITUTO FEDERAL
Baiano

Structs

Encapsulamento: dados encapsulados em uma entidade

Acoplamento: código depende de um tipo único (struct Pessoa)

```c
struct Pessoa{
    char nome[50], endereco[50];
    int id, idade;
    char sexo;
};

int main() {
    struct Pessoa p1, p2, p3, p4;

    return 0;
}
```

```c
int main() {
    char nomeP1[50], nomeP2[50], nomeP3[50], nomeP4[50];
    char enderecoP1[50], enderecoP2[50], enderecoP3[50], enderecoP4[50];
    int idP1, idP2, idP3, idP4;
    int idadeP1, idadeP2, idadeP3, idadeP4;
    char sexoP1, sexoP2, sexoP3, sexoP4;

    return 0;
}
```

INSTITUTO FEDERAL
Baiano

Structs

INSTITUTO FEDERAL
Baiano

- Cada campo (variável) de uma struct pode ser acessado utilizando o operador ponto (.).

```c
#include <stdio.h>
#include <string.h>

struct Pessoa{
    char nome[50], endereco[50];
    int id, idade;
    char sexo;
};
```

```c
int main() {
    struct Pessoa p;
    p.id = 1;

    printf("Digite um nome: ");
    fgets(p.nome, sizeof(p.nome), stdin);
    printf("Digite um endereço: ");
    fgets(p.endereco, sizeof(p.endereco), stdin);
    printf("Digite uma idade: ");
    scanf("%d", &p.idade);
    printf("Digite um sexo: ");
    scanf(" %c", &p.sexo);

    return 0;
}
```

Structs

- Pode-se definir várias estruturas diferentes dentro do mesmo arquivo .c.
- Estruturas distintas podem possuir campos com o mesmo nome, pois cada campo pertence apenas à sua própria estrutura.

```c
#include <stdio.h>

struct Ponto2D{
    int x, y;
};

struct Ponto3D{
    int x, y, z;
};

int main(){
    struct Ponto2D p1;
    struct Ponto3D p2;
    p1.x = 10;
    p2.x = 12;
    printf("p1.x = %d \n", p1.x);
    printf("p2.x = %d \n", p2.x);
    return 0;
}
```

Structs

INSTITUTO FEDERAL
Baiano

- É possível fornecer uma lista de valores para inicializar uma estrutura, usando uma sintaxe semelhante à inicialização de vetores.
- Os valores são atribuídos aos campos na mesma ordem em que eles foram declarados na struct.
- Campos que não forem explicitamente inicializados recebem o valor padrão do tipo (zero para tipos numéricos e '\0' para caracteres).

Structs

INSTITUTO FEDERAL
Baiano

```c
#include <stdio.h>

struct Pessoa{
    char nome[50], endereco[50];
    int id, idade;
    char sexo;
};

int main(){
    struct Pessoa p1 = {"Lucas", "Rua 1", 1, 18, 'M'};
    struct Pessoa p2 = {"Maria", "Rua 2"};

    return 0;
}
```

Structs

INSTITUTO FEDERAL
Baiano

```c
#include <stdio.h>

struct Pessoa{
    char nome[50], endereco[50];
    int id, idade;
    char sexo;
};

int main(){
    struct Pessoa p1 = {"Lucas", "Rua 1", 1, 18, 'M'};
    struct Pessoa p2 = {"Maria", "Rua 2"};

    return 0;
}
```

id = 0
idade = 0
sexo = '\0'

Structs

INSTITUTO FEDERAL
Baiano

- Estruturas em C permitem atribuição direta entre variáveis, desde que sejam do mesmo tipo.
- Isso significa que podemos copiar todos os campos de uma struct para outra usando o operador de atribuição (=).

```c
struct Pessoa{
    char nome[50], endereco[50];
    int id, idade;
    char sexo;
};
```

```c
int main(){
    struct Pessoa p1 = {"Lucas", "Rua 1", 1, 18, 'M'};
    struct Pessoa p2 = {"Maria", "Rua 2"};
    struct Pessoa p3 = p1;

    return 0;
}
```

Structs

```
struct Ponto2D{
    int x, y;
};

struct NovoPonto2D{
    int x, y;
};

int main(){
    struct Ponto2D p1 = {1, 2};
    struct NovoPonto2D p2 = {2, 3};
    p1 = p2;
    return 0;
}
```

INSTITUTO FEDERAL
Baiano

Structs

```
struct Ponto2D{
    int x, y;
};

struct NovoPonto2D{
    int x, y;
};

int main(){
    struct Ponto2D p1 = {1, 2};
    struct NovoPonto2D p2 = {2, 3};
    p1 = p2;
    return 0;
}
```

INSTITUTO FEDERAL
Baiano

Structs

- Como a struct é um tipo de dado, a linguagem C também permite declarar vetores (arrays) de estruturas.

```c
struct Pessoa{
    char nome[50], endereco[50];
    int id, idade;
    char sexo;
};
```

```c
int main(){
    struct Pessoa pessoas[5];
    struct Pessoa p1 = {"Lucas", "Rua 1", 1, 18, 'M'};
    pessoas[0] = p1;

    strcpy(pessoas[1].nome, "Maria");
    strcpy(pessoas[1].endereco, "Rua 2");
    pessoas[1].id = 2;
    pessoas[1].idade = 20;
    pessoas[1].sexo = 'F';

    return 0;
}
```

INSTITUTO FEDERAL
Baiano

Structs

```c
struct Pessoa{
    char nome[50], endereco[50];
    int id, idade;
    char sexo;
};
```

```c
int main(){
    struct Pessoa pessoas[5];

    for (int i = 0; i < 5; i++) {
        printf("\nPessoa %d\n", i + 1);
        printf("Digite o nome: ");
        fgets(pessoas[i].nome, sizeof(pessoas[i].nome), stdin);
        printf("Digite o endereco: ");
        fgets(pessoas[i].endereco, sizeof(pessoas[i].endereco), stdin);
        printf("Digite o id: ");
        scanf("%d", &pessoas[i].id);
        printf("Digite a idade: ");
        scanf("%d", &pessoas[i].idade);
        printf("Digite o sexo (M/F): ");
        scanf(" %c", &pessoas[i].sexo);
        getchar();
    }

    return 0;
}
```

INSTITUTO FEDERAL
Baiano

Structs

INSTITUTO FEDERAL
Baiano

- Os campos de uma estrutura podem ser passados para funções de duas formas:
  - Por valor: A função recebe uma cópia do valor. Alterações feitas dentro da função não afetam o valor original.
  - Por referência: A função recebe o endereço da variável (ponteiro). Assim, é possível modificar o valor original dentro da função.

Structs

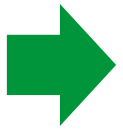
```c
void imprimeValorInteiro(int valor){
    printf("Valor = %d\n", valor);
}

void imprimeReferenciaInteiro(int *referencia){
    printf("Valor = %d\n", *referencia);
}

int main(){
    struct Pessoa p1 = {"Lucas", "Rua 1", 1, 18, 'M'};
    imprimeValorInteiro(p1.idade);
    imprimeReferenciaInteiro(&p1.idade);

    return 0;
}
```

INSTITUTO FEDERAL
Baiano

Structs

INSTITUTO FEDERAL
Baiano

```c
void imprimeValorInteiro(int valor){
    printf("Valor = %d\n", valor);
}

void imprimeReferenciaInteiro(int *referencia){
    printf("Valor = %d\n", *referencia);
}

int main(){
    struct Pessoa p1 = {"Lucas", "Rua 1", 1, 18, 'M'};
    imprimeValorInteiro(p1.idade);
    imprimeReferenciaInteiro(&p1.idade);

    return 0;
}
```



```
Valor = 18
Valor = 18
```

Structs

```c
void modificaValorInteiro(int valor){
    valor = valor + 1;
}

void modificaReferenciaInteiro(int *referencia){
    *referencia = *referencia + 1;
}

int main(){
    struct Pessoa p1 = {"Lucas", "Rua 1", 1, 18, 'M'};
    printf("%d\n", p1.idade);
    modificaValorInteiro(p1.idade);
    printf("%d\n", p1.idade);
    modificaReferenciaInteiro(&p1.idade);
    printf("%d\n", p1.idade);

    return 0;
}
```

INSTITUTO FEDERAL
Baiano

Structs

```c
void modificaValorInteiro(int valor){
    valor = valor + 1;
}

void modificaReferenciaInteiro(int *referencia){
    *referencia = *referencia + 1;
}

int main(){
    struct Pessoa p1 = {"Lucas", "Rua 1", 1, 18, 'M'};
    printf("%d\n", p1.idade);
    modificaValorInteiro(p1.idade);
    printf("%d\n", p1.idade);
    modificaReferenciaInteiro(&p1.idade);
    printf("%d\n", p1.idade);

    return 0;
}
```

```
18
18
19
```

INSTITUTO FEDERAL
Baiano

Structs

INSTITUTO FEDERAL
Baiano

- Uma estrutura também pode ser passada para funções de duas formas:
  - Por valor: A função recebe uma cópia completa da estrutura.
    - Qualquer modificação feita dentro da função não altera a estrutura original.
    - Útil quando a função apenas consulta os dados.
  - Por referência: A função recebe o endereço da estrutura (ponteiro para struct).
    - Assim, é possível modificar diretamente os dados da estrutura original.
    - Mais eficiente para estruturas grandes e necessário quando a função precisa alterar os dados.

Structs

```c
void imprimeStructValor(struct Pessoa p){
    printf("[Nome = %s, ", p.nome);
    printf("endereço = %s, ", p.endereco);
    printf("id = %d, ", p.id);
    printf("idade = %d, ", p.idade);
    printf("sexo = %c]\n", p.sexo);
}

int main(){
    struct Pessoa p1 = {"Lucas", "Rua 1", 1, 18, 'M'};
    imprimeStructValor(p1);

    return 0;
}
```

```
[Nome = Lucas, endereço = Rua 1, id = 1, idade = 18, sexo = M]
```

INSTITUTO FEDERAL
Baiano

Structs

```c
void imprimeStructReferencia(struct Pessoa *p){
    printf("[Nome = %s, ", (*p).nome);
    printf("endereço = %s, ", (*p).endereco);
    printf("id = %d, ", (*p).id);
    printf("idade = %d, ", (*p).idade);
    printf("sexo = %c]\n", (*p).sexo);
}

int main(){
    struct Pessoa p1 = {"Lucas", "Rua 1", 1, 18, 'M'};
    imprimeStructReferencia(&p1);

    return 0;
}
```

```
[Nome = Lucas, endereço = Rua 1, id = 1, idade = 18, sexo = M]
```

INSTITUTO FEDERAL
Baiano

Structs

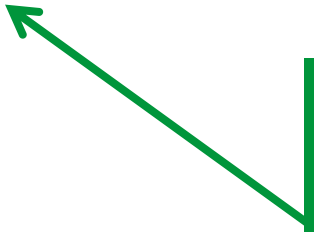
```c
void imprimeStructReferencia(struct Pessoa *p){
    printf("[Nome = %s, ", p->nome);
    printf("endereço = %s, ", p->endereco);
    printf("id = %d, ", p->id);
    printf("idade = %d, ", p->idade);
    printf("sexo = %c]\n", p->sexo);
}

int main(){
    struct Pessoa p1 = {"Lucas", "Rua 1", 1, 18, 'M'};
    imprimeStructReferencia(&p1);

    return 0;
}
```

```
[Nome = Lucas, endereço = Rua 1, id = 1, idade = 18, sexo = M]
```

Structs

INSTITUTO FEDERAL
Baiano

- O comando typedef permite ao programador criar um novo nome (apelido) para um tipo de dado já existente.
- Sintaxe geral: typedef tipoExistente novoNome;



novoNome pode ser usado no programa como se fosse um tipo comum da linguagem.

Structs

INSTITUTO FEDERAL
Baiano

- O comando typedef permite ao programador criar um novo nome (apelido) para um tipo de dado já existente.
- Sintaxe geral: typedef tipoExistente novoNome;

```c
int main(){
    typedef int Inteiro;
    Inteiro x = 4;
    printf("Valor de x = %d \n", x);

    return 0;
}
```

novoNome pode ser usado no programa como se fosse um tipo comum da linguagem.

Structs

INSTITUTO FEDERAL
Baiano

- O comando typedef também pode ser utilizado junto com estruturas, permitindo criar um novo nome para o tipo definido pela struct.
    - Isso evita a necessidade de escrever struct toda vez que formos declarar variáveis desse tipo.

```c
struct Pessoa{
    char nome[50], endereco[50];
    int id, idade;
    char sexo;
};

int main(){
    struct Pessoa p1;

    return 0;
}
```

```c
typedef struct{
    char nome[50], endereco[50];
    int id, idade;
    char sexo;
} Pessoa;

int main(){
    Pessoa p1;

    return 0;
}
```

Exercícios

INSTITUTO FEDERAL
Baiano

1. Implemente um programa que leia do teclado o nome, a idade e o endereço de uma pessoa e armazene esses dados em uma estrutura.
2. Modifique o programa anterior para armazenar os dados de 5 pessoas em um vetor estático de estruturas.
3. Evolua o programa para armazenar as 5 pessoas em um vetor dinâmico de estruturas (alocado com malloc).
4. Realoque o vetor dinâmico para permitir o armazenamento de mais 2 pessoas, utilizando realloc.

Exercícios

INSTITUTO FEDERAL
Baiano

5. Implemente:
    a) Uma função que receba uma estrutura por valor e imprima os dados da pessoa;
    b) Outra função que receba uma estrutura por referência (ponteiro) e também imprima os dados.

Dúvidas

INSTITUTO FEDERAL
Baiano

ESTRUTURA DE DADOS

Curso de Licenciatura em Ciências da Computação

Lucas Sampaio Leite