

Linguagem C: Estruturas básicas definidas pelo usuário

Sumário

- Estruturas;
 - Referenciando elementos de estruturas;
 - Matrizes de estruturas;
 - Matrizes e estruturas dentro de estruturas;

Estruturas básicas definidas pelo usuário

- A linguagem C permite criar tipos de dados definidos pelo usuário de cinco formas diferentes;
 - **Estruturas:** agrupamento de variáveis sob um nome;
 - **Campo de bits:** variação da estrutura que permite fácil acesso aos bits dentro de uma palavra;
 - **União:** permite que a mesma porção da memória seja definida por dois ou mais tipos diferentes de variáveis;
 - **Enumeração:** lista de símbolos;
 - **typedef:** define um novo nome para um tipo existente;

Programa sem estruturas

- Ainda com nossas limitações, como faríamos um programa para cadastrar clientes (nome, cpf, rg, endereço, cidade e estado) em um estabelecimento comercial qualquer?
 - Declaração das variáveis;
 - Apresentação do programa;
 - Leitura dos dados e;
 - Impressão dos valores para confirmação.

- Vejamos uma sugestão para esse programa:

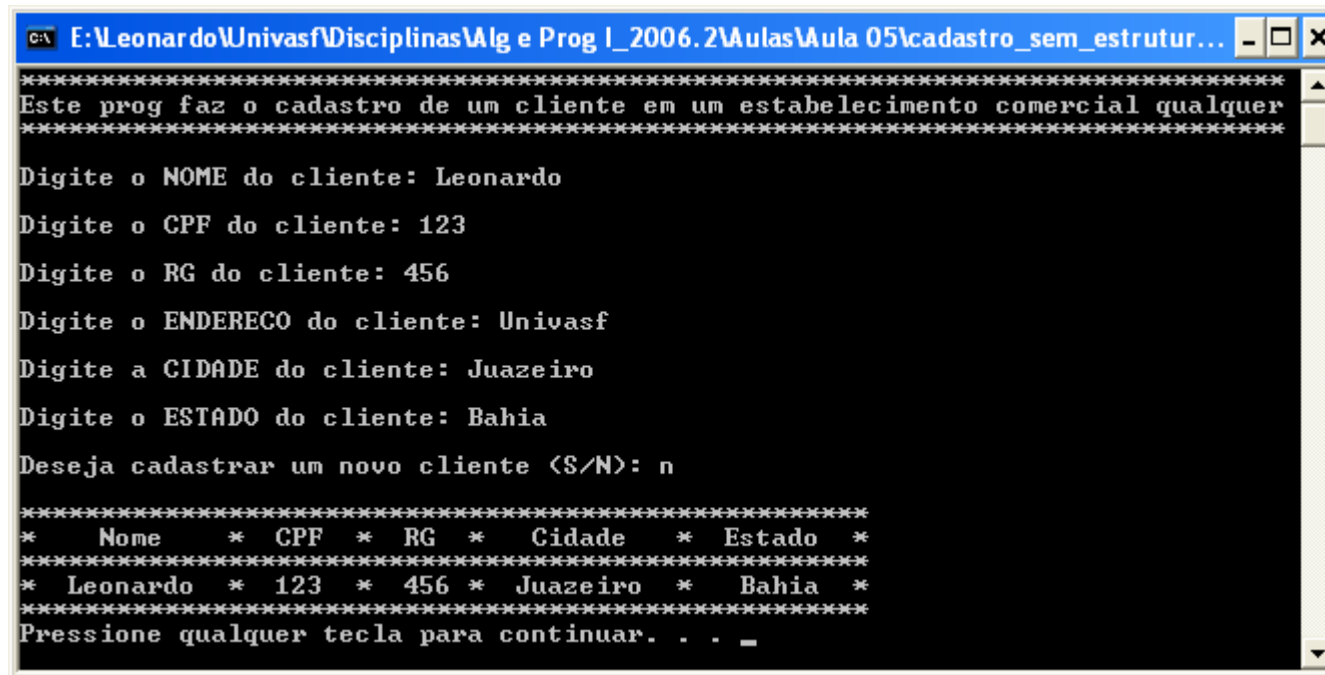
Programa sem estruturas

```
#include<stdio.h>
/* Este programa faz o cadastro de um cliente SEM estruturas */
int main()
{
    char nome[30];
    char cpf[15];
    char rg[12];
    char endereco[50];
    char cidade[20];
    char estado[15];
    char op;
    do{
        system("cls");
        printf("*****");
        printf("\nEste prog faz o cadastro de um cliente em um estabelecimento comercial qualquer");
        printf("\n*****");
        printf("\n\nDigite o NOME do cliente: ");    gets(nome);
        printf("\nDigite o CPF do cliente: ");    gets(cpf);
        printf("\nDigite o RG do cliente: ");    gets(rg);
        printf("\nDigite o ENDEREÇO do cliente: "); gets(endereco);
        printf("\nDigite a CIDADE do cliente: "); gets(cidade);
        printf("\nDigite o ESTADO do cliente: "); gets(estado);
        printf("\nDeseja cadastrar um novo cliente (S/N): "); op = getche();
    }while( (op!='n') &&(op!='N') );
    printf("\n\n*****");
    printf("\n*      Nome      * CPF * RG *      Cidade      * Estado *");
    printf("\n*****");
    printf("\n*   %s   *   %s   *   %s   *   %s   *   %s   *", nome, cpf, rg, cidade, estado);
    printf("\n*****\n");
    system("pause");
    return 0;
}
```

Limpa a tela

Programa sem estruturas

- A saída no console para o programa anterior será:



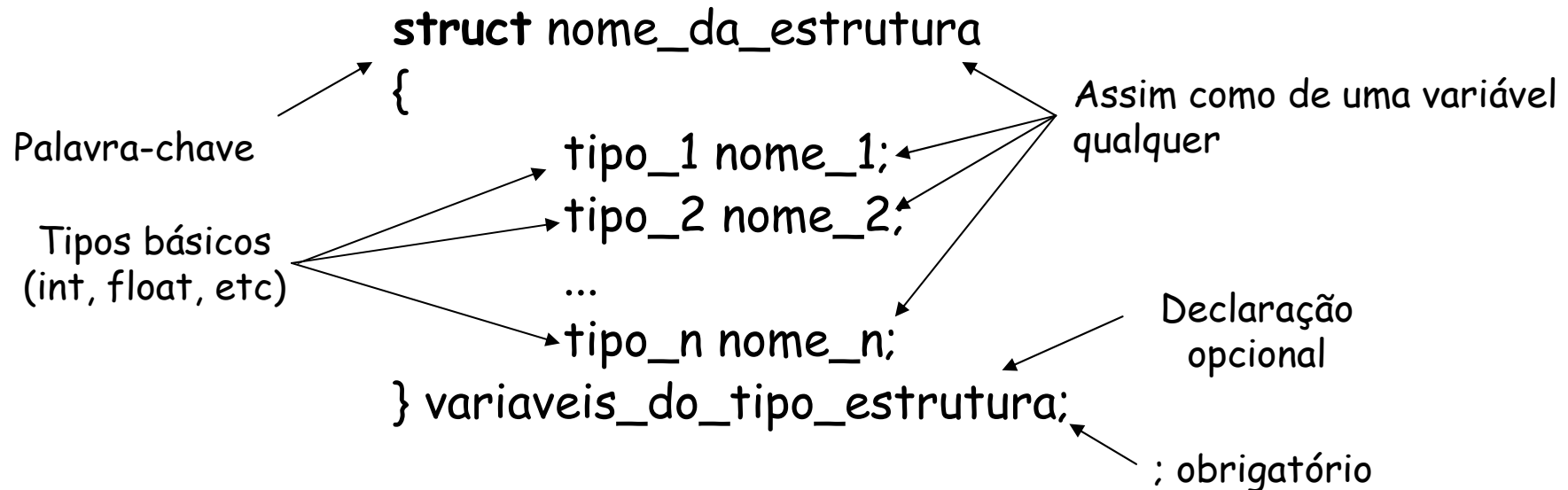
```
E:\Leonardo\Univasf\Disciplinas\Alg e Prog I_2006.2\Aulas\Aula 05\cadastro_sem_estrutur...
*****
Este prog faz o cadastro de um cliente em um estabelecimento comercial qualquer
*****

Digite o NOME do cliente: Leonardo
Digite o CPF do cliente: 123
Digite o RG do cliente: 456
Digite o ENDERECO do cliente: Univasf
Digite a CIDADE do cliente: Juazeiro
Digite o ESTADO do cliente: Bahia
Deseja cadastrar um novo cliente (S/N): n

*****
*  Nome  * CPF * RG * Cidade * Estado *
*****
* Leonardo * 123 * 456 * Juazeiro * Bahia *
*****
Pressione qualquer tecla para continuar. . . _
```

Estruturas

- Em C, uma estrutura é uma coleção de variáveis referenciadas inicialmente por um nome;
- As variáveis que compreendem a estrutura são chamadas membros da estrutura, ou simplesmente, elementos ou campos da estrutura;
- A forma geral de uma estrutura em C é a seguinte:



Estruturas

- O fragmento de código a seguir mostra como criar um modelo de estrutura que define os campos de cadastro do cliente:

```
#include<stdio.h>
#include<string.h>
/* Este programa faz o cadastro de um cliente COM estruturas */
struct cliente ← Tipo de dados
{
    char nome[30];
    char cpf[15];
    char rg[12];
    char endereco[50];
    char cidade[20];
    char estado[15];
};
int main()
{
    struct cliente c; ← c é uma variável do tipo cliente
    char op;
    do{
        system("cls");
```


Estruturas

- Afinal de contas o que é cliente?
 - Cliente é uma **ESTRUTURA DE DADOS**
- A estrutura de dados chamada cliente pode ter suas variáveis criadas também da seguinte forma:

```
#include<stdio.h>
#include<string.h>
/* Este programa faz o cadastro de um cliente COM estruturas */
struct cliente
{
    char nome[30];
    char cpf[15];
    char rg[12];
    char endereco[50];
    char cidade[20];
    char estado[15];
} a,b,c;
int main()
{
    char op;
    do{
        system("cls");
```

----- Criação de 3 estruturas de dados (a, b e c)

Referenciando elementos de estruturas

- Elementos individuais de estruturas são referenciados por meio de operadores (**operador ponto**);
- A **forma geral** para acessar um elemento de estrutura é:

`nome_da_estrutura.nome_do_elemento`

- Assim para atribuir uma string ao nome de cliente e imprimir na tela seu conteúdo, teríamos:
`gets(c.nome);`
`printf("%s", c.nome);`

Referenciando elementos de estruturas

```
#include<stdio.h>
#include<string.h>
/* Este programa faz o cadastro de um cliente COM estruturas */
struct cliente
{
    char nome[30];
    char cpf[15];
    char rg[12];
    char endereco[50];
    char cidade[20];
    char estado[15];
};
int main()
{
    struct cliente a,b,c;
    char op;
    do{
        system("cls");
        printf("*****");
        printf("\nEste prog faz o cadastro de um cliente em um estabelecimento comercial qualquer");
        printf("\n*****");
        printf("\n\nDigite o NOME do cliente: ");    gets(c.nome);
        printf("\nDigite o CPF do cliente: ");    gets(c.cpf);
        printf("\nDigite o RG do cliente: ");    gets(c.rg);
        printf("\nDigite o ENDEREÇO do cliente: ");    gets(c.endereco);
        printf("\nDigite a CIDADE do cliente: ");    gets(c.cidade);
        printf("\nDigite o ESTADO do cliente: ");    gets(c.estado);
        printf("\nDeseja cadastrar um novo cliente (S/N): ");    op = getche();
    }while((op!='n') &&(op!='N'));
    printf("\n\n*****");
    printf("\n*      Nome      * CPF * RG * Cidade * Estado *");
    printf("\n*****");
    printf("\n*   %s   * %s * %s * %s * %s *", c.nome, c.cpf, c.rg, c.cidade, c.estado);
    printf("\n*****\n");
    system("pause");
    return 0;
}
```

Definição da estrutura pode vir antes da função principal

Declaração de mais de uma estrutura

Manipulação dos dados da estrutura

Matrizes de estruturas

- Até o momento nosso cadastro de cliente só inicializa uma única estrutura;
- A solução é... declarar uma matriz de estruturas, vejamos:

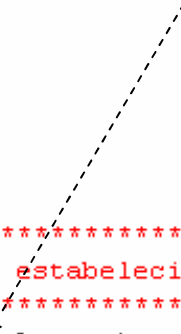
```
#include<stdio.h>
#include<string.h>
/* Este programa faz o cadastro de um cliente COM estruturas */
struct cliente
{
    char nome[30];
    char cpf[15];
    char rg[12];
    char endereco[50];
    char cidade[20];
    char estado[15];
};
int main()
{
    struct cliente c[20];
    char op;
    int cont=0, n;
    do{
        system("cls");
```

Declaração de uma matriz
de estruturas com 20
elementos do tipo cliente.

Matrizes de estruturas

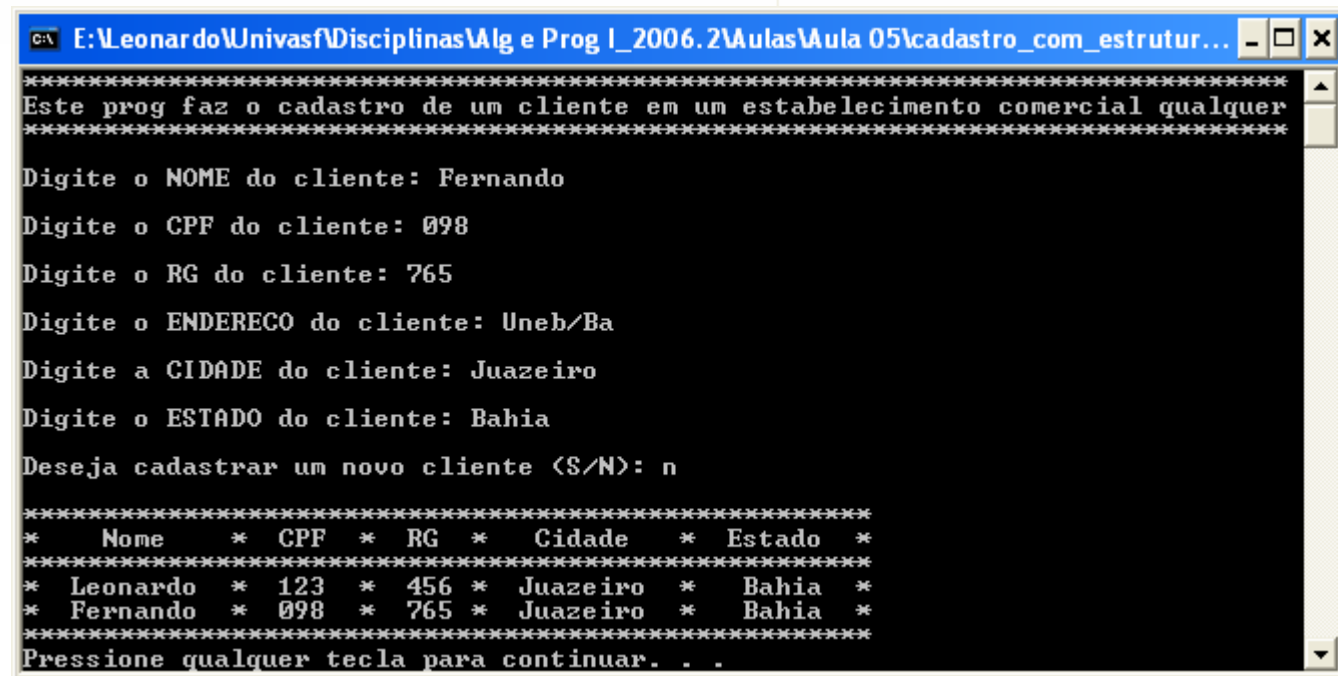
```
#include<stdio.h>
#include<string.h>
/* Este programa faz o cadastro de um cliente COM estruturas */
struct cliente
{
    char nome[30];
    char cpf[15];
    char rg[12];
    char endereco[50];
    char cidade[20];
    char estado[15];
};
int main()
{
    struct cliente c[20];
    char op;
    int cont=0, n;
    do{
        system("cls");
        printf("*****");
        printf("\nEste prog faz o cadastro de um cliente em um estabelecimento comercial qualquer");
        printf("\n*****");
        printf("\n\nDigite o NOME do cliente: ");    gets(c[cont].nome);
        printf("\nDigite o CPF do cliente: ");    gets(c[cont].cpf);
        printf("\nDigite o RG do cliente: ");    gets(c[cont].rg);
        printf("\nDigite o ENDEREÇO do cliente: "); gets(c[cont].endereco);
        printf("\nDigite a CIDADE do cliente: ");    gets(c[cont].cidade);
        printf("\nDigite o ESTADO do cliente: ");    gets(c[cont].estado);
        printf("\nDeseja cadastrar um novo cliente (S/N): ");    op = getche();
        cont++;
    }while ( (op!='n') &&(op!='N') );
```

Basta alterar a referencia ao índice das estruturas



Matrizes de estruturas

```
n = cont;
printf("\n\n*****");
printf("\n*   Nome   * CPF * RG * Cidade * Estado *");
printf("\n*****");
for(cont=0; cont<n; cont++)
printf("\n* %s * %s * %s * %s * %s *", c[cont].nome, c[cont].cpf, c[cont].rg, c[cont].cidade, c[cont].estado);
printf("\n*****\n");
system("pause");
return 0;
}
```



```
*****
Este prog faz o cadastro de um cliente em um estabelecimento comercial qualquer
*****

Digite o NOME do cliente: Fernando
Digite o CPF do cliente: 098
Digite o RG do cliente: 765
Digite o ENDERECO do cliente: Uneb/Ba
Digite a CIDADE do cliente: Juazeiro
Digite o ESTADO do cliente: Bahia
Deseja cadastrar um novo cliente (S/N): n

*****
*   Nome   * CPF * RG * Cidade * Estado *
*****
* Leonardo * 123 * 456 * Juazeiro * Bahia *
* Fernando * 098 * 765 * Juazeiro * Bahia *
*****
Pressione qualquer tecla para continuar. . .
```

Matrizes e estruturas dentro de estruturas

- Agora vejamos um problema até agora não pensado:

```
#include<stdio.h>
#include<string.h>
/* Este programa faz o cadastro de um cliente COM estruturas */
struct cliente
{
    char nome[30];
    char cpf[15];
    char rg[12];
    char endereco[50];
    char cidade[20];
    char estado[15];
};
int main()
{
    struct cliente c[20];
    char op;
    int cont=0, n;
    do{
        system("cls");
```

← Temos um campo "divisível"

Rua

Número

Bairro

Cep

etc

O que fazer?

Matrizes e estruturas dentro de estruturas

- Criar uma estrutura dentro de outra estrutura, vejamos:

Criação da estrutura tipoEndereco dentro da estrutura cliente

```
#include<string.h>
/* Este programa faz o cadastro de um cliente com estruturas aninhadas*/
struct tipoEndereco
{
    char rua[30];
    char num[3];
    char bairro[15];
    char cep[9];
};
struct cliente
{
    char nome[30];
    char cpf[15];
    char rg[12];
    char cidade[20];
    char estado[15];
    struct tipoEndereco endereco;
};
int main()
{
    struct cliente c[20];
    char op;
    int cont=0, n;
    do{
        system("cls");
```

Estrutura deve ser definida antes de ser utilizada

Declaração da estrutura cliente

Matrizes e estruturas dentro de estruturas

- Para referenciar os elementos de cada estrutura partimos das estruturas mais externas, vejamos:

```
int main()
{
    struct cliente c[20];
    char op;
    int cont=0, n;
    do{
        system("cls");
        printf("*****");
        printf("\nEste prog faz o cadastro de um cliente em um estabelecimento comercial qualquer");
        printf("\n*****");
        printf("\n\nDigite o NOME do cliente: ");    gets(c[cont].nome);
        printf("\nDigite o CPF do cliente: ");    gets(c[cont].cpf);
        printf("\nDigite o RG do cliente: ");    gets(c[cont].rg);
        printf("\nDigite a RUA do cliente: ");    gets(c[cont].endereco.rua);
        printf("\nDigite a CIDADE do cliente: ");    gets(c[cont].cidade);
        printf("\nDigite o ESTADO do cliente: ");    gets(c[cont].estado);
        printf("\nDeseja cadastrar um novo cliente (S/N): ");    op = getche();
        cont++;
    }while ( (op!='n') && (op!='N') );
```

Endereco é um campo da estrutura c
(do tipo cliente). Endereco é, ainda,
uma estrutura que contém o campo rua



Bibliografia

- SCHILDT H. *"C Completo e Total"*, Makron Books. SP, 1997.
- UFMG "Curso de Linguagem C", Universidade Federal de Minas Gerais.