

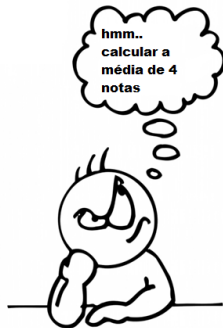
BCC 201 - Introdução à Programação I

Estruturas (registros)

Guillermo Cámara-Chávez
UFOP

Estruturas (*struct*) I

- ▶ Imaginemos que queremos ler as notas de 4 provas para um aluno e
- ▶ calcular a média do aluno



Estruturas (*struct*) II

Nome	Nota1	Nota2	Nota3	Nota4	Média
Pedro	5.6	6.0	7.3	5.6	6.1

Estruturas (*struct*) III



Estruturas (*struct*) IV

Aluno

Nome	Nota1	Nota2	Nota3	Nota4	Média
Pedro	5.6	6.0	7.3	5.6	6.1



Estruturas (*struct*) V

- ▶ “*structs*” são Estruturas de Dados Heterogêneas
- ▶ Uma estrutura **agrupa várias** variáveis **numa só**.
- ▶ Funciona como uma espécie de “ficha”.
- ▶ A ficha armazena diversos **dados relacionados**, porém de **tipos diferente**.

Estruturas (*struct*) VI

- ▶ A estrutura, então, serve para **agrupar um conjunto de dados não similares, formando um novo tipo** de dados.
- ▶ As estruturas podem conter elementos de qualquer tipo de dados tais como `int`, `char`, `float`, `double`, ponteiros, vetores, matrizes, *strings* ou mesmo outras estruturas.

Estruturas (*struct*) VII

► Declaração

```
struct nome_da_estrutura
{
    tipo_campo1 nome_campo1;
    tipo_campo2 nome_campo2;
    ...
};
```

onde :

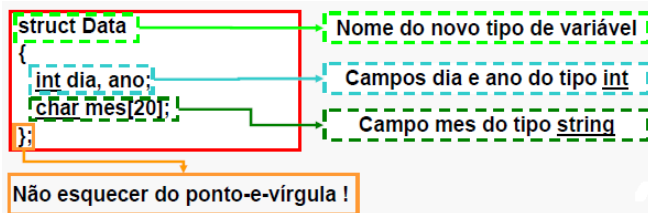
- nome_da_estrutura é o nome do tipo da estrutura e, entre chaves,
- tipo_campo1, tipo_campo2, ... é a lista com os tipos de dados em C (char, int, float, double, char[])

Estruturas (*struct*) VIII

- ▶ Exemplo 1:
 - ▶ Definir um novo tipo de variável Data (`struct Data`)
 - ▶ A partir daquele momento o compilador passa a conhecer um outro tipo de dado, chamado `struct Data`
 - ▶ Dita estrutura está composta por dois inteiros e um vetor de caracteres (inteiros: dia e ano, string: mes)

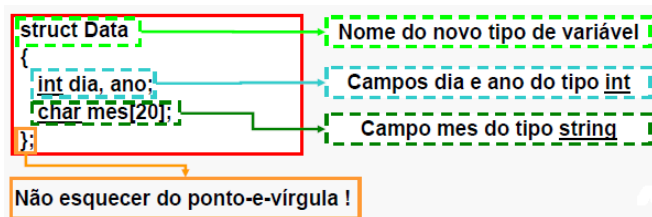
Estruturas (*struct*) IX

- ▶ Definir um novo tipo de variável Data (`struct Data`)
- ▶ A partir daquele momento o compilador passa a conhecer um outro tipo de dado, chamado `struct Data`
- ▶ Dita estrutura está composta por dois inteiros e um vetor de caracteres (inteiros: dia e ano, string: mes)



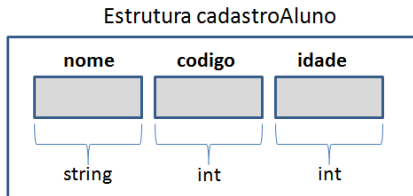
Estruturas (*struct*) X

- ▶ Data não é uma variável, senão o nome pelo que é conhecido um **novo tipo de dados**
- ▶ Cada um dos elementos de Data é denominado campo.



Estruturas (*struct*) XI

► Exemplo 2

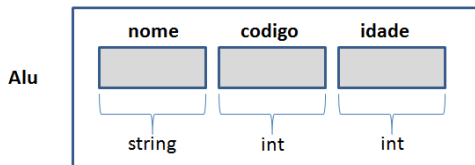


Para atribuir valores a seus campos fazemos diretamente inserindo um "." (ponto) entre o nome da variável e o campo que nos interessa.

Estruturas (*struct*) XII

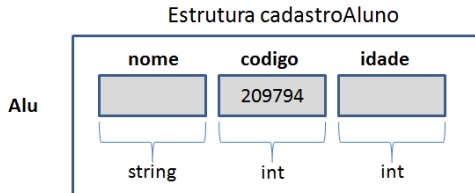
```
struct cadastroAluno Alu;
```

Estrutura cadastroAluno



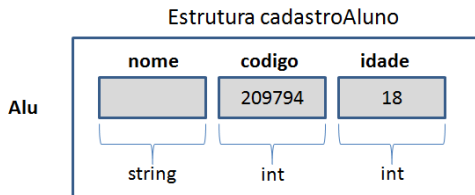
Estruturas (*struct*) XIII

```
Alu.codigo = 209794;
```



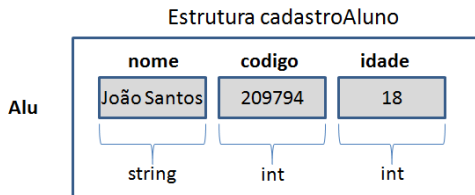
Estruturas (*struct*) XIV

Alu.idade = 18;



Estruturas (*struct*) XV

```
strcpy(Alu.nome, "Joao Santos");
```



Estruturas (*struct*) XVI

```
struct cadastroAluno
{
    char nome[50];
    int  codigo;
    int  idade;
};
```

Estruturas (*struct*) XVII

```
int main()
{
    struct cadastroAluno Alu1, Alu2;
    strcpy(Alu1.nome, "João Santos");
    Alu1.codigo = 365833;
    Alu1.idade = 19;

    printf("Inserir Nome: ");
    fgets(Alu2.nome, 50, stdin);
    printf("Inserir codigo e idade: ")
    scanf("%d %d", &Alu2.codigo, &Alu2.idade);

    printf("%s %d %d \n", Alu1.nome,
            Alu1.codigo, Alu1.idade);
    printf("%s %d %d \n", Alu2.nome,
            Alu2.codigo, Alu2.idade);
    return 0;
}
```

Vetor de Registros I

- ▶ Declaração:

```
struct nome_registro nome_variavel[tamanho_vetor];
```

- ▶ Uso:

```
nome_variavel[indice].nome_do_campo;
```

Vetor de Registros II

Criar o programa que permita cadastrar 20 alunos.

Vetor de Registros III

```
struct cadastroAluno{
    char nome[50];
    int  codigo;
    int  idade;
};
int main(){
    struct cadastroAluno A[20];
    int i;
    for (i = 0; i < 20; i++){
        printf("Inserir dados aluno %d", i+1);
        printf("Digite nome:");
        fgets(A[i].nome, 50, stdin);
        printf("Digite codigo e idade:");
        scanf("%d %d", &A[i].codigo, &A[i].idade);
    }
    for (i = 0; i < 20; i++)
        printf("%s %d %d \n", A[i].nome, A[i].codigo,
            A[i].idade);

    return 0;
}
```

Exemplos I

Inserir uma lista de n filmes. A estrutura para os filmes deve conter os seguintes campos: nome e o ano de lançamento.

Exemplos II

Filme

Nome	Ano
<input type="text"/>	<input type="text"/>
char	int

Exemplos III

	Filme				
	<table><tr><th>Nome</th><th>Ano</th></tr><tr><td>Avatar</td><td>2009</td></tr></table>	Nome	Ano	Avatar	2009
Nome	Ano				
Avatar	2009				
F1					
	Filme				
	<table><tr><th>Nome</th><th>Ano</th></tr><tr><td>Crash</td><td>2005</td></tr></table>	Nome	Ano	Crash	2005
Nome	Ano				
Crash	2005				
F2					
	...				
	Filme				
	<table><tr><th>Nome</th><th>Ano</th></tr><tr><td>Gladiator</td><td>2000</td></tr></table>	Nome	Ano	Gladiator	2000
Nome	Ano				
Gladiator	2000				
Fn					

Exemplos IV

0	<table><tr><th colspan="2">Filme</th></tr><tr><th>Nome</th><th>Ano</th></tr><tr><td>Avatar</td><td>2009</td></tr></table>	Filme		Nome	Ano	Avatar	2009
Filme							
Nome	Ano						
Avatar	2009						
1	<table><tr><th colspan="2">Filme</th></tr><tr><th>Nome</th><th>Ano</th></tr><tr><td>Crash</td><td>2005</td></tr></table>	Filme		Nome	Ano	Crash	2005
Filme							
Nome	Ano						
Crash	2005						
	...						
n	<table><tr><th colspan="2">Filme</th></tr><tr><th>Nome</th><th>Ano</th></tr><tr><td>Gladiator</td><td>2000</td></tr></table>	Filme		Nome	Ano	Gladiator	2000
Filme							
Nome	Ano						
Gladiator	2000						

Exemplos V

```
struct filme{
    char nome[50];
    int ano;
};
int main(){
    int n;
    struct filme f[100];
    do{
        printf("Inserir número de filmes ");
        scanf("%n", &n);
    }while (n > 100 || n < 0);
    //Insere n nomes de filmes
    for (int i = 0; i < n; i++)
    {
        printf("Nome e ano: \n");
        fgets(f[i].nome, 50, stdin);
        scanf("%d", &f[i].ano);
    }
    // ... (continua)
```

Exemplos VI

```
...  
//Mostra os filmes inseridos  
for (int i = 0; i < n; i++)  
{  
    printf("%s %d \n", f[i].nome, f[i].ano);  
}  
return 0;  
}
```

Exemplos VII

Inicialização: é possível inicializar uma variável do tipo estrutura

```
struct filme
{
    char nome[50];
    int ano;
};
int main()
{
    struct filme f = {"Avatar", 2009};
    printf("Filme: %s Ano: %d", f.nome, f.ano);
    return 0;
}
```

Exemplos VIII

Criar uma estrutura aluno que deve conter os seguintes campos: nome, idade, identidade. Criar um vetor de 10 elementos e logo inserir valores nele. Mostrar os dados do aluno mais novo.

Exemplos IX

```
struct Aluno{
    char nome[50];
    int idade;
    int id;
};
int main()
{
    struct Aluno lista[10];
    int menor, pos;
    for (int i = 0; i < 10; i++)
    {
        printf("Inserir dados do aluno %d \n", i+1);
        fgets(lista[i].nome, 50, stdin);
        scanf("%d %d", &lista[i].idade, &lista[i].id);
    }
    ...
}
```

Exemplos X

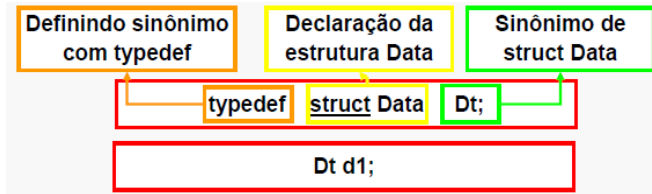
```
...
pos = 0;
menor = lista[0].idade;
for (int i = 1; i < 10; i++)
    if (lista[i].idade < menor)
    {
        menor = lista[i].idade;
        pos = i;
    }
printf("Aluno mais novo: %s %d %d",
       lista[pos].nome, lista[pos].idade,
       lista[pos].id);
return 0;
}
```

Novos Tipos: *typedef* I

- ▶ A declaração de uma variável do tipo estrutura emprega a palavra reservada `struct`

```
struct Data d1;
```

- ▶ A palavra reservada `typedef` estabelece um sinônimo para qualquer conjunto de palavras



Novos Tipos: *typedef* II

- ▶ Outra alternativa é empregar a palavra reservada `typedef` ao mesmo tempo em que é definida a estrutura:

The diagram shows a C code snippet enclosed in a blue border. The code defines a structure named `Data` and uses `typedef` to create an alias `Dt` for it. The code is as follows:

```
typedef struct Data
{
    int Dia, Ano;
    char Mes[20];
} Dt;

main()
{
    Dt d1;
    d1.Dia = 26;
    d1.Mes = "Jan";
    d1.Ano = 93; }
```

Two red boxes with arrows point to specific parts of the code, accompanied by explanatory text:

- A red box highlights `typedef` and `struct Data`. An arrow points from this box to a text box that reads: "Typedef define que o sinônimo de struct Data {...} é a palavra Dt."
- A red box highlights `Dt d1;` in the `main()` function. An arrow points from this box to a text box that reads: "Não é necessário empregar a palavra struct Data e sim apenas Dt."

Novos Tipos: *typedef* III

- As três formas possíveis para declarar um novo tipo Data com ou sem typedef são:

<pre><u>struct</u> Data { <u>int</u> Dia, Ano; <u>char</u> Mes[20]; }; main() { <u>struct</u> Data d1; d1.Dia = 26; d1.Mes = "Jan"; d1.Ano = 93; }</pre>	<pre><u>struct</u> Data { <u>int</u> Dia, Ano; <u>char</u> Mes[20]; }; typedef <u>struct</u> Data Dt; main() { Dt d1; d1.Dia = 26; d1.Mes = "Jan"; d1.Ano = 93; }</pre>	<pre><u>typedef</u> <u>struct</u> Data { <u>int</u> Dia, Ano; <u>char</u> Mes[20]; } Dt; main() { Dt d1; d1.Dia = 26; d1.Mes = "Jan"; d1.Ano = 93; }</pre>
---	---	---

Estruturas Compostas I

Declaração de estrutura composta

```
struct nome_do_tipo_da_estrutura
{
    struct nome_de_outra_estrutura nome_campo1;
    tipo_campo2 nome_campo2;
    ...
};
```

Desta forma, uma estrutura pode ser parte de outra estrutura!

Estruturas Compostas II

Exemplo

Criar uma ficha cadastral de alunos. Essa lista deve contar a seguinte informação: nome, identidade e data de nascimento;

Estruturas Compostas III

```
struct data
{
    int dia, mes, ano;
};
struct ficha_cadastral
{
    char nome[50];
    int id;
    struct data nascimento;
};
```

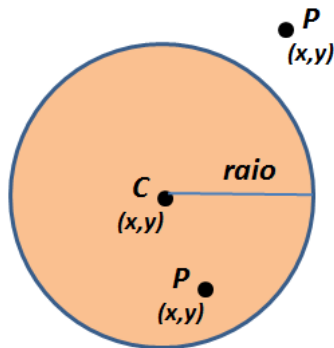
Estruturas Compostas IV

```
int main()
{
    struct ficha_cadastral alunos;
    printf("Inserir nome: \n");
    fgets(alunos.nome, 50, stdin);
    printf("Inserir identidade: \n");
    scanf("%d", &alunos.id);
    printf("Inserir data de nascimento (dd/mm/aa) \n");
    scanf("%d %d %d", &alunos.nascimento.dia,
                    &alunos.nascimento.mes,
                    &alunos.nascimento.ano);

    return 0;
}
```

Estruturas Compostas V

Definir uma estrutura círculo, essa estrutura deve conter a seguinte informação: ponto de origem (criar estrutura ponto) e raio. Logo, inserir um ponto e responde se esse ponto está dentro do círculo. (criar uma função distancia que receba como dados de entrada os dois pontos)



Estruturas Compostas VI

```
struct Ponto{
    int x, y;
};
struct Circulo{
    struct Ponto C;
    double raio;
};
double distancia(struct Ponto, struct Ponto);
```


Estruturas Compostas VII

```
int main()
{
    struct Circulo circ;
    Ponto P;
    double d
    printf("Inserir dados do circulo: \n");
    printf("Coordenadas em x e y: \n");
    scanf("%d %d", &circ.C.x, &circ.C.y) ;
    printf("Raio: \n");
    scanf("%lf", &circ.raio);
    printf("Inserir ponto P (x,y) \n");
    scanf("%d %d", &P.x, &P.y);
    ...
}
```

Estruturas Compostas VIII

```
...
d = distancia (circ.C, P);
if (d <= circ.Raio)
    printf("ponto dentro do circulo \n");
else
    printf("ponto fora do circulo \n");
return 0;
}

double distancia(Ponto A, Ponto B)
{
    double d;
    d = sqrt( pow(A.x - B.x, 2) + pow(A.y - B.y, 2) );
    return d;
}
```

Exercícios Propostos I

1. Criar um programa que permita ao usuário digitar os dados de 3 alunos, (seus nomes, idades e registros acadêmicos). Em seguida, pedir ao usuário para digitar uma idade, e o programa deve imprimir os nomes e idades dos alunos com idade menor que a digitada.
2. Foi realizada uma pesquisa entre 500 habitantes de uma certa região. De cada habitante foram coletados os dados: idade, sexo, salário e número de filhos. Crie a estrutura de dados adequada para armazenar estas informações e faça uma função que armazene as informações digitadas pelo usuário na estrutura de dados criada. Faça também uma função que calcula a média do salário dos habitantes.

FIM