

ESTRUTURA DE DADOS: ESTRUTURAS DEFINIDAS PELO PROGRAMADOR

Prof. Dr. Jean Nunes

OPERATOR CLASSES

types.Operator:
X mirror to the select
object.mirror_mirror_x"
for X"

ATRIBUIÇÃO ENTRE ESTRUTURAS

- Atribuições entre estruturas só podem ser feitas quando as estruturas possuem o mesmo nome!

```
struct cadastro c1, c2;  
c1 = c2; //CORRETO
```

```
struct cadastro c1;  
struct ficha c2;  
c1 = c2; //ERRADO!! TIPOS DIFERENTES
```

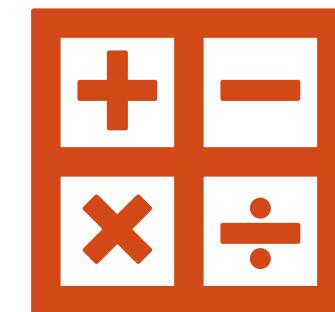


ATRIBUIÇÃO ENTRE ESTRUTURAS



```
struct cadastro c[10];  
c[1] = c[2]; //CORRETO
```

Em arrays, a atribuição entre diferentes elementos do array é válida!



Note que nesse caso, os tipos dos diferentes elementos do array são sempre IGUAIS.



ESTRUTURAS DE ESTRUTURAS

Sendo uma estrutura um tipo de dado, podemos declarar uma estrutura que utilize outra estrutura previamente definida:

```
struct endereco{  
    char rua[50]  
    int numero;  
};  
  
struct cadastro{  
    char nome[50];  
    int idade;  
    struct endereco ender;  
};
```

char nome[50];
int idade;
struct endereco ender
char rua[50];
int numero;

cadastro



ESTRUTURAS DE ESTRUTURAS

Nesse caso, o acesso aos dados do **endereço** do cadastro é feito utilizando novamente o operador ponto “.”.

```
struct cadastro c;  
  
//leitura  
gets (c.nome);  
scanf ("%d", &c.idade);  
gets (c.ende.rua);  
scanf ("%d", & c.ende.numero);  
  
//atribuição  
strcpy (c.nome, "João");  
c.idade = 34;  
strcpy (c.ende.rua, "Avenida 1");  
c.ende.numero = 131;
```

```
struct ponto {  
    int x, y;  
};  
  
struct retangulo {  
    struct ponto inicio, fim;  
};  
  
struct retangulo r = {{10, 20}, {30, 40}};
```

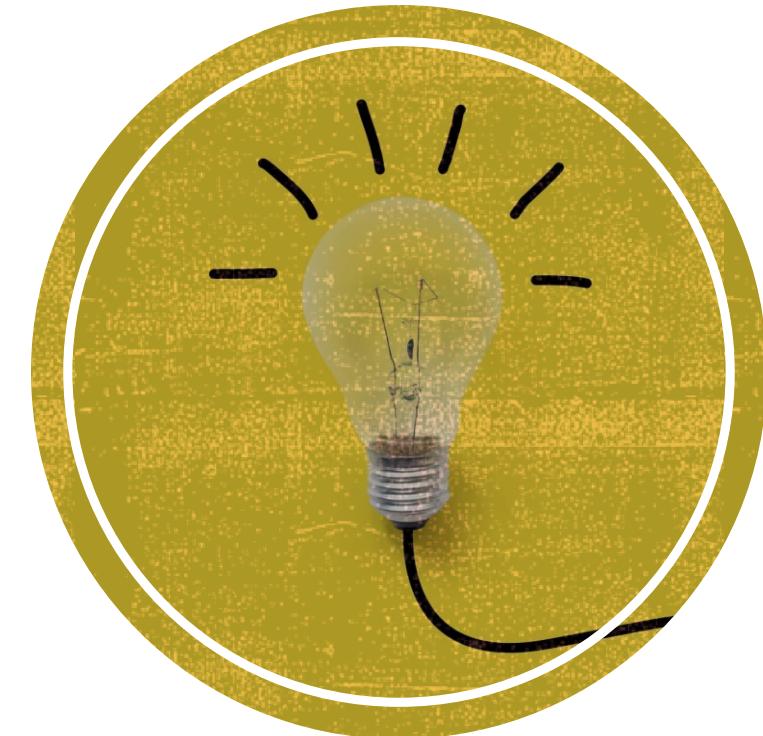
ESTRUTURAS DE ESTRUTURAS

Inicialização de uma estrutura de estruturas



TYPEDEF

- A linguagem C permite que o programador defina os seus próprios tipos com base em outros tipos de dados existentes.



```
typedef tipo_existente novo_nome;
```



TYPEDEF

```
#include <stdio.h>
#include <stdlib.h>

typedef int inteiro;

int main() {
    int x = 10;
    inteiro y = 20;
    y = y + x;
    printf("Soma = %d\n", y);

    return 0;
}
```

O comando **typedef** não cria um novo tipo chamado **inteiro**. Ele apenas cria um sinônimo (**inteiro**) para o tipo **int**



TYPEDEF

```
struct cadastro{  
    char nome[300];  
    int idade;  
};  
// redefinindo o tipo struct cadastro  
typedef struct cadastro CadAlunos;  
  
int main(){  
    struct cadastro alunol;  
    CadAlunos aluno2;  
  
    return 0;  
}
```

É muito utilizado para definir nomes mais simples para estrutura, evitando carregar a palavra **struct** sempre que referenciamos a estrutura



1. CONTROLE DE NOTAS

I. Usando struct(s), escreva um programa que represente o cadastro de notas de alunos da disciplina de estrutura de dados a partir das seguintes fórmulas expressas pelas Equações abaixo:

O algoritmo deve ler as notas das atividades práticas (AP1, AP2 e AP3) e a nota da prova (NP) considerando as duas avaliações (AV1 e AV2) e, em seguida, calcular a nota final (NF) de cada aluno e o *status* de aprovação.

$$NF = \frac{(AV_1 + AV_2)}{2}, \quad 0 \leq NF \leq 10$$

$$AV_1 = \left(\left(\frac{AP_1 + AP_2 + AP_3}{3} \right) * 0,3 \right) + (NP * 0,7)$$

$$AV_2 = \left(\left(\frac{AP_1 + AP_2 + AP_3}{3} \right) * 0,3 \right) + (NP * 0,7)$$

II. Faça o cadastro de 10 alunos e mostre o resultado.

