

# 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"

# VARIÁVEIS



Categorias de variáveis:

simples: **int, float, double e char**; compostas homogêneas (ou seja, do mesmo tipo): definidas por **array**.



A linguagem C permite a criação de novas estruturas a partir dos tipos básicos.

**struct**



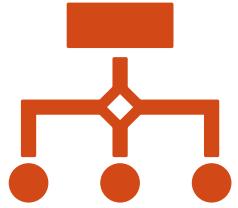
# ESTRUTURAS

```
struct nomestruct{  
    tipo1 campo1;  
    tipo2 campo2;  
    ...  
    tipoN campoN;  
};
```

Uma estrutura pode ser vista como um **novo tipo de dado**, que é formado por composição de variáveis de outros tipos.



# ESTRUTURAS



**Agrupamento de dados.**

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

char nome[50]
int idade;
char rua[50];
int numero;

cadastro



**Ex.: cadastro de pessoas.**

Todas essas informações são da mesma pessoa, logo podemos agrupá-las.

Isso facilita também lidar com dados de outras pessoas no mesmo programa



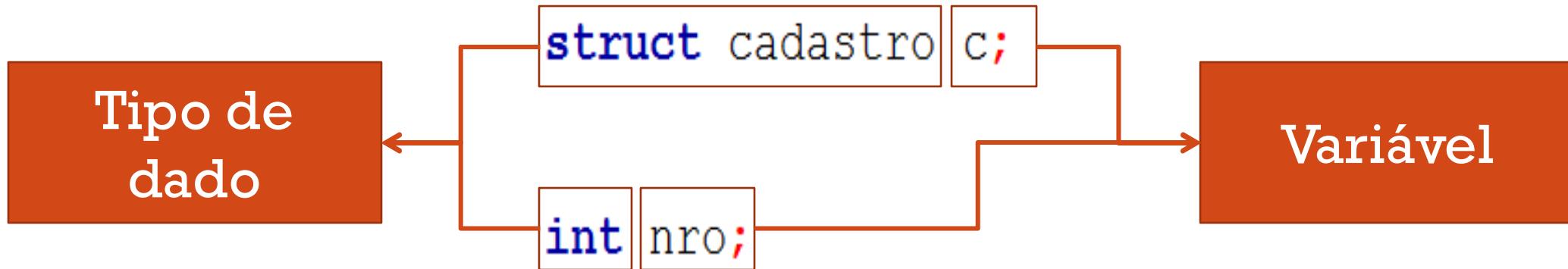
```
struct cadastro c;
```

Uma vez definida a estrutura, uma **variável** pode ser declarada de modo similar aos tipos já existentes.

---

## ESTRUTURAS - DECLARAÇÃO





- Por ser um tipo definido pelo programador, usa-se a palavra **struct** antes do tipo da nova variável.

---

## ESTRUTURAS - DECLARAÇÃO



# **EXERCÍCIO**

**Declare uma estrutura capaz de armazenar a matrícula e 3 notas para um dado aluno e, em seguida, calcule e armazene a média.**



# EXERCÍCIO - SOLUÇÃO

```
struct aluno {  
    int matricula;  
    int nota1, nota2, nota3;  
};  
  
struct aluno {  
    int matricula;  
    int nota1;  
    int nota2;  
    int nota3;  
};  
  
struct aluno {  
    int matricula;  
    int nota[3];  
};
```



# ESTRUTURAS

O uso de estruturas facilita na manipulação dos dados do programa. Imagine declarar 4 cadastros, para 4 pessoas diferentes:

```
char nome1[50], nome2[50], nome3[50], nome4[50];
int idade1, idade2, idade3, idade4;
char rua1[50], rua2[50], rua3[50], rua4[50]
int numero1, numero2, numero3, numero4;
```



# ESTRUTURAS

## Declarando a estrutura

```
struct cadastro{  
    char nome[50];  
    int idade;  
    char rua[50];  
    int numero;  
};  
  
//declarando 4 cadastros  
struct cadastro c1, c2, c3, c4
```



```
//declarando a variável  
struct cadastro c;  
  
//acessando os seus campos  
strcpy(c.nome, "João");  
scanf("%d", &c.idade);  
strcpy(c.rua, "Avenida 1");  
c.numero = 1082;
```

Cada variável da estrutura pode ser acessada com o operador ponto “.”

---

## ACESSO ÀS VARIÁVEIS

Como é feito o acesso às variáveis da estrutura?



# ACESSO ÀS VARIÁVEIS

Como nos arrays, uma estrutura pode ser previamente inicializada:

```
struct ponto {  
    int x;  
    int y;  
};
```

```
struct ponto p1 = { 220, 110 };
```



```
struct cadastro c;  
  
gets(c.nome); //string  
scanf("%d", &c.idade); //int  
gets(c.rua); //string  
scanf("%d", &c.numero); //int
```

Lemos cada variável independentemente, respeitando seus tipos.

## ACESSO ÀS VARIÁVEIS

E se quiséssemos ler os valores das variáveis da estrutura?





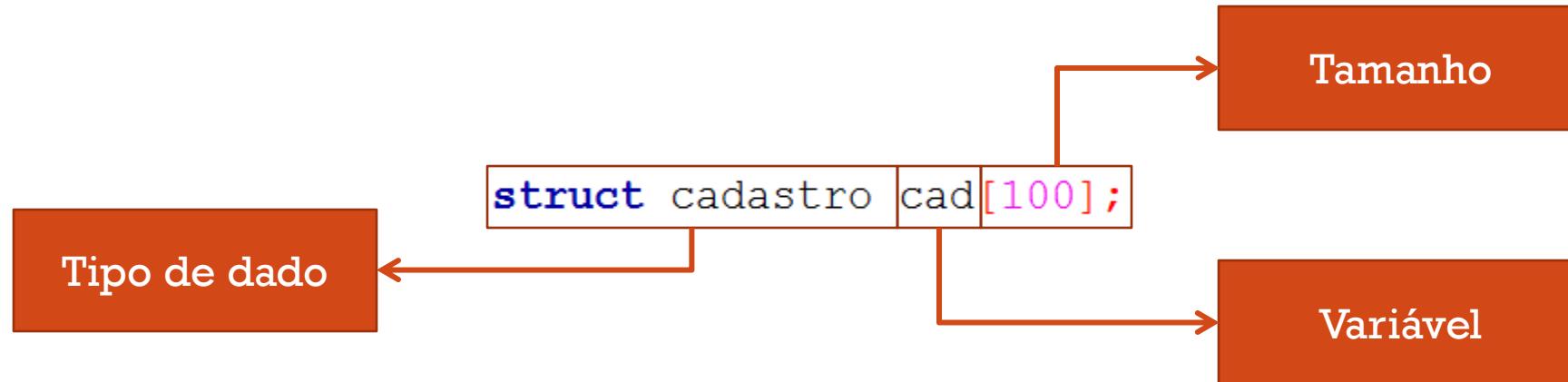
# ESTRUTURAS

E se, ao invés de 4  
cadastros, quisermos  
fazer 100 cadastros?



# ARRAY DE ESTRUTURAS

- Sua declaração é similar a declaração de um array de um tipo básico



- Desse modo, declara-se um array de 100 posições, onde cada posição é do tipo **struct cadastro**.



# ARRAY DE ESTRUTURAS

## ATENÇÃO:

- **struct**: define um “conjunto” de variáveis que podem ser de tipos diferentes;
- **array**: é uma “lista” de elementos de mesmo tipo.

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

char nome[50];	char nome[50];	char nome[50];	char nome[50];
int idade;	int idade;	int idade;	int idade;
char rua[50]	char rua[50]	char rua[50]	char rua[50]
int numero;	int numero;	int numero;	int numero;

cad[0]

cad[1]

cad[2]

cad[3]



# ARRAY DE ESTRUTURAS

```
int main() {  
    struct cadastro c[4];  
    int i;  
    for(i=0; i<4; i++) {  
        gets(c[i].nome);  
        scanf("%d", &c[i].idade);  
        gets(c[i].rua);  
        scanf("%d", &c[i].numero);  
    }  
    system("pause");  
    return 0;  
}
```

Num array de estruturas, o operador de ponto (.) vem depois dos colchetes ([ ]) do índice do **array**.



# EXERCÍCIO

Utilizando a estrutura abaixo, faça um programa para ler a matrícula e as 3 notas calculando a média de 10 alunos.

```
struct aluno {  
    int matricula;  
    float nota1, nota2, nota3;  
    float media;  
};
```



# **EXERCÍCIO - SOLUÇÃO**

Utilizando a estrutura abaixo, faça um programa para ler a matrícula e as 3 notas de 10 alunos e imprima os dados.

