

Programação Estruturada - Resumo das Aulas

Professor : Rogério Malheiros dos Santos

Aula 0 - Explicação do Uso da Linguagem C para o aprendizado

Entrar no site TIOBE com tabela de linguagens de programação

O C está em primeiro ou segundo lugar e 6 a 7 das principais linguagens tem sintaxe parecida com o C

No mercado :

Java para Desktop, Web e Mobile

C# para Desktop , Web e Mobile (Xamarin)

Python para Desktop, Web (em Crescimento)

Aula 1 - Revisão de Estruturas de Repetição

A Linguagem C trabalha com 3 estruturas de repetição :

. for

. while

.do...while

Aula 1 - Revisão de Estruturas de Repetição

A Estrutura for é a mais complexa das três pois se divide em quatro partes :

Parte 1- Inicialização da Variável Contadora : onde se dá o valor inicial da variável tipo inteira denominada Contadora (Obrigatória no FOR)

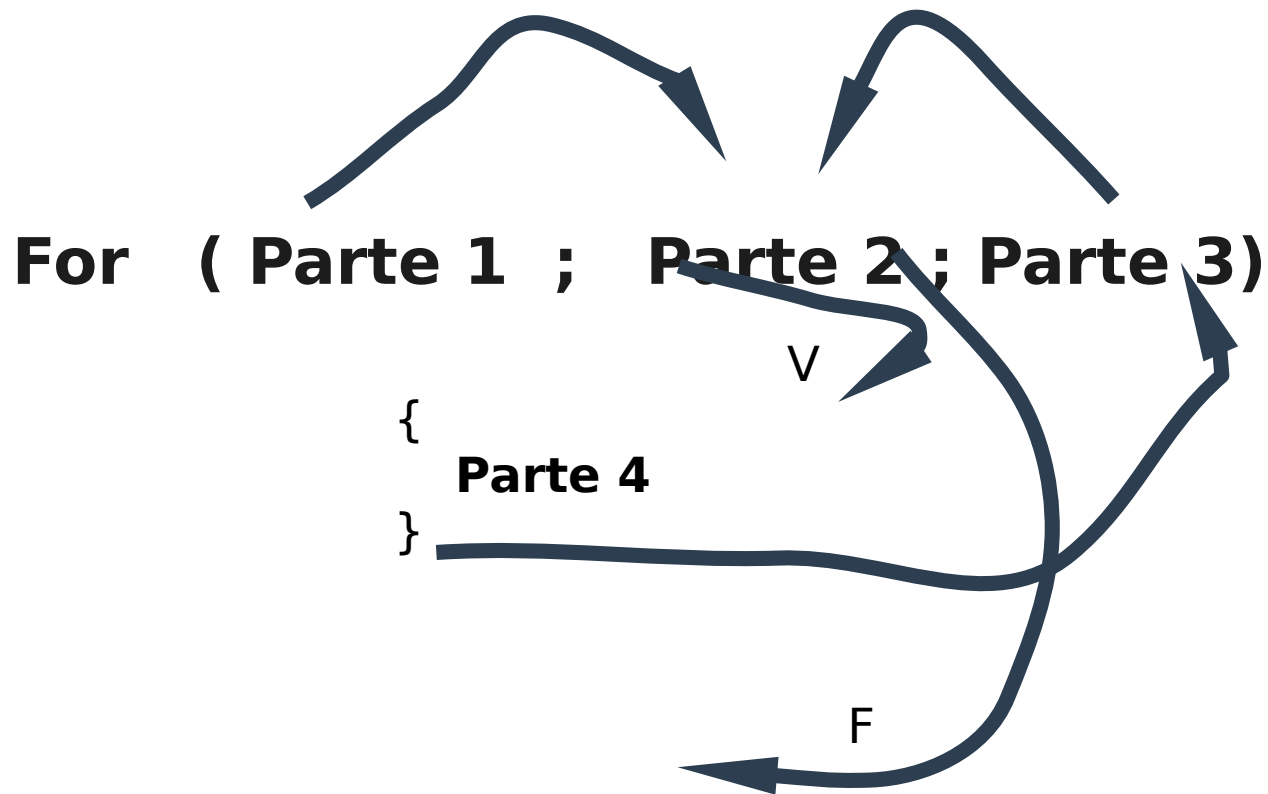
Parte 2 - Teste de Repetição : Expressão Booleana que se for verdadeira executa o bloco do FOR

Parte 3 - Incremento : Onde se faz uma modificação da variável Contadora pelas operações de * + - /

Parte 4 - Bloco FOR : Comandos - Quando são dois comandos ou mais é obrigatório delimitar por { }

Aula 1 - Revisão de Estruturas de Repetição

Sintaxe e Fluxo



Aula 1 - Revisão de Estruturas de Repetição

Exemplo

Rodar Programa FOR que está no AVA

Aula 1 - Revisão de Estruturas de Repetição

A Estrutura While possui duas partes executadas nesta ordem

Parte 1 - Teste de Repetição : Expressão Booleana que se for verdadeira executa o Bloco While

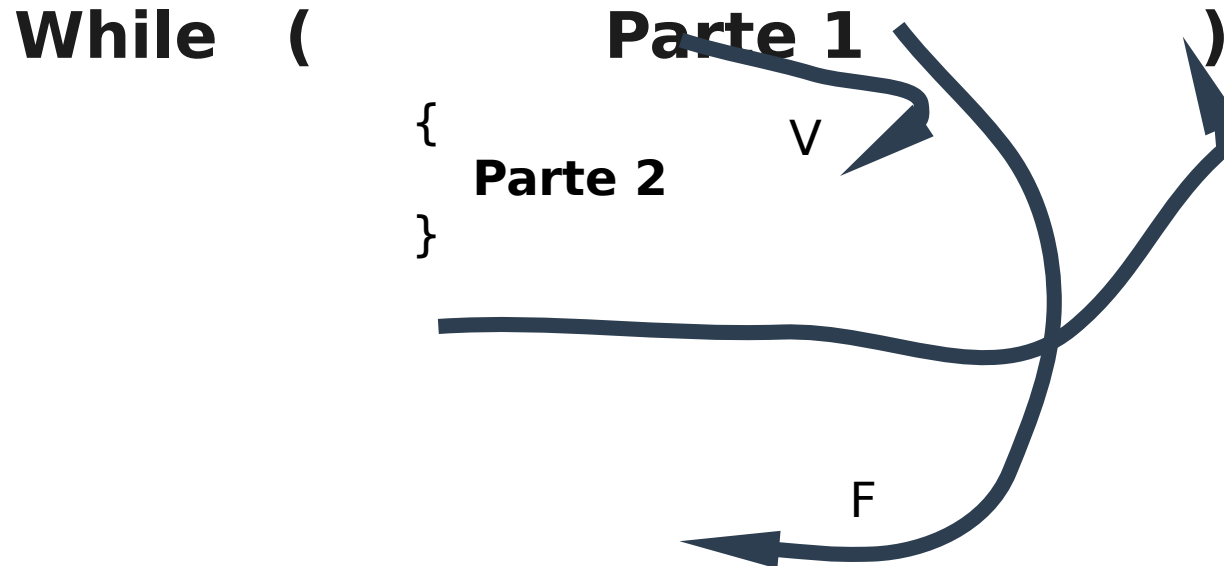
Parte 2 - Execução : Bloco While - Comandos

Teste é feito no início da Estrutura

Consequência : Bloco While pode nem ser executado

Aula 1 - Revisão de Estruturas de Repetição

Sintaxe e Fluxo



Aula 1 - Revisão de Estruturas de Repetição

Exemplo

Rodar Programa while que está no AVA

Aula 1 - Revisão de Estruturas de Repetição

A Estrutura Do While possui duas partes executadas nesta ordem

Parte 1 - Execução : Bloco Do While - Comandos

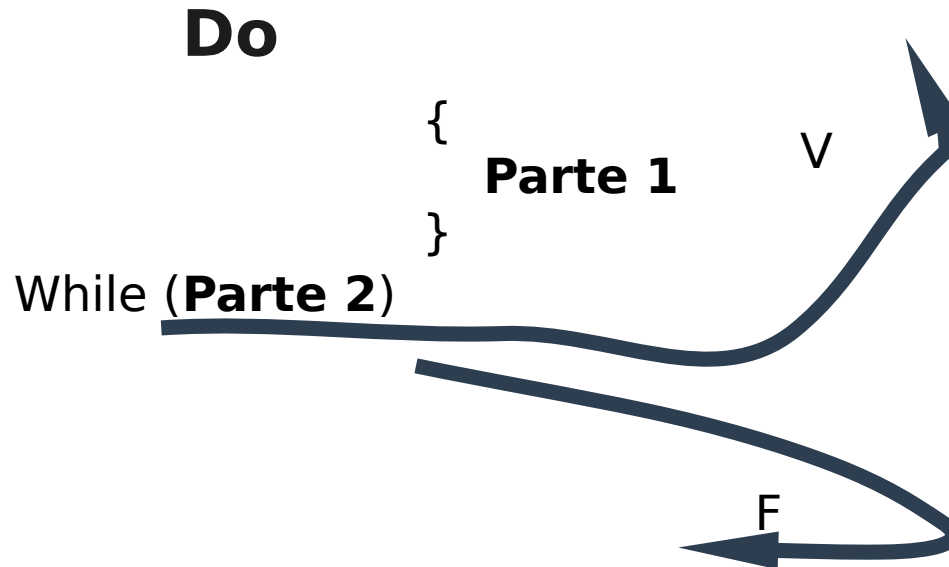
Parte 2 - Teste de Repetição : Expressão Booleana que se for verdadeira reexecuta o bloco Do While

Teste de Repetição é feito no final da Estrutura

Consequência : Bloco Do While é executado pelo menos uma vez

Aula 1 - Revisão de Estruturas de Repetição

Sintaxe e Fluxo



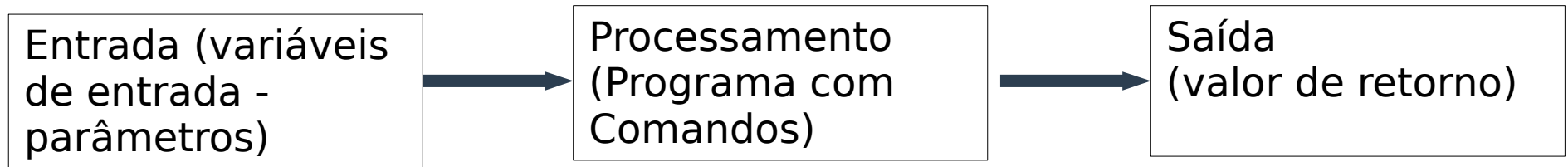
Aula 1 - Revisão de Estruturas de Repetição

Exemplo

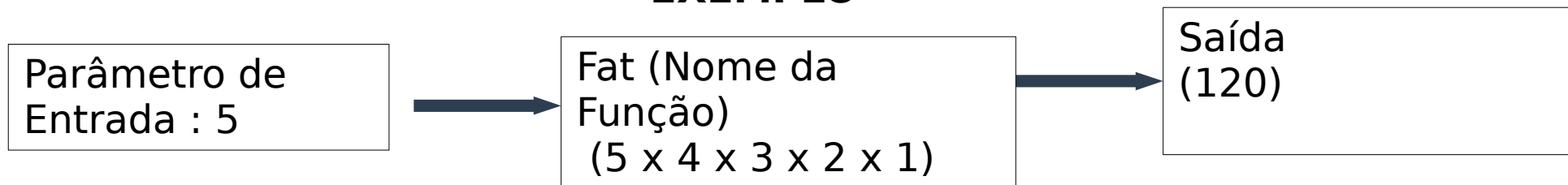
Rodar Programa do-while que está no AVA

Aula 2 - Funções em C

Toda a função trabalha da maneira abaixo



EXEMPLO



Na prática: `fat (5) =120`

Aula 2 - Funções em C

A função no C é definida antes do programa principal

SINTAXE

```
Tipo_Saída Nome_Função(Tipo_Entrada parâmetro )  
{  
  Processamento (programa) acabando  
  obrigatoriamente com a linha abaixo  
  
  return valor de retorno;  
}
```

Aula 2 - Funções em C

Exemplo

Rodar Programa da função fatorial fat
(AVA)

Rodar programa da função potência pot
(AVA)

Aula 2 - Funções em C

Exercício

Criar um programa que leia os números inteiros positivos A,B e C e calcule

$$\text{FAT}(A) + \text{POT}(B,C)$$

Aula 3 – Funções Recursivas em C

Uma Função $F(N)$ (N inteiro) é Recursiva quando é definida da maneira abaixo

$$F(N) = \begin{cases} M & \text{Quando } N=p \text{ (Condição Base)} \\ G(F) & \text{Quando } N>p \end{cases}$$

Onde M é um número inteiro dado

E p é o menor valor de N na função também dado (M e p são Base)

E $G(F)$ é uma fórmula que chama a própria função F (recursividade)

Aula 3 – Funções Recursivas em C

EXEMPLO : CALCULE F(5) para as função abaixo

$$a)F(N)=\begin{cases} 1 \text{ Quando } N=0 \\ 2F(N-1) \text{ Quando } N>0 \end{cases}$$

$$F(0)=1$$

$$F(1) = 2F(0) = 2 \times 1 = 2$$

$$F(2) = 2F(1) = 2 \times 2 = 4$$

$$F(3) = 2F(2) = 2 \times 4 = 8$$

$$F(4) = 2F(3) = 2 \times 8 = 16$$

$$**F(5) = 2F(4) = 2 \times 16 = 32**$$

Obs : Esta função é a forma recorrente da Função

$$F(N) = 2^N$$

Aula 3 – Funções Recursivas em C

EXERCÍCIO : CALCULE F(5) para as funções abaixo

$$a)F(N)=\begin{cases} 5 & \text{Quando } N=1 \\ F(N-1) + 10 & \text{Quando } N>1 \end{cases}$$

$$b)F(N)=\begin{cases} 1 & \text{Quando } N=1 \\ F(N-1) + N^2 & \text{Quando } N>1 \end{cases}$$

$$c)F(N)=\begin{cases} 1 & \text{Quando } N=1 \text{ ou } N=2 \\ F(N-1) + F(N-2) & \text{Quando } N>2 \end{cases}$$

Aula 3 – Funções Recursivas em C

Uma Função $F(N)$ (N inteiro) é Recursiva quando é definida da maneira abaixo

$$F(N) = \begin{cases} M & \text{Quando } N=p \text{ (Condição Base)} \\ G(F) & \text{Quando } N>p \end{cases}$$

Onde M é um número inteiro dado

E p é o menor valor de N na função também dado (M e p são Base)

E $G(F)$ é uma fórmula que chama a própria função F (recursividade)

Aula 3 – Funções Recursivas em C

A função no C é definida antes do programa principal

SINTAXE

```
Tipo_Saída Nome_Função(Tipo_Entrada parâmetro )  
{  
  Processamento (programa)  
  
  If (parâmetro == p) return M  
    Else return G(F)  
}
```

Aula 3 – Funções Recursivas em C

Exemplo : Considere a função $F(N)$ dada por

$$F(N) = \begin{cases} 1 & \text{Quando } N=0 \\ 2F(N-1) & \text{Quando } N>0 \end{cases}$$

Aula 3 – Funções Recursivas em C

A função no C é definida antes do programa principal

SINTAXE

```
int F(int N )  
{  
    if (N == 0) return 1  
  
    else return 2*F(N-1);  
}
```

Aula 3 – Funções Recursivas em C

Exemplo : Função FAT (fatorial)
(FAT(0)=1)

FAT(5) = 5 X 4 X 3 X 2 X 1 . Por sua vez :

FAT(4) = 4 x 3 x 2 x 1 Assim podemos escrever

$$\mathbf{FAT(5) = 5 \times FAT(4)}$$

Da mesma maneira

FAT(4) = 4 x 3 x 2 x 1

FAT(3) = 3 X 2 X 1

Logo **FAT(4) = 4xFAT(3)**

Assim para $N > 0$

$$\mathbf{FAT(N) = N \times FAT(N-1)}$$

Aula 3 – Funções Recursivas em C

A função FAT(N) então pode ser escrita na seguinte forma recursiva

$$\text{FAT}(N) = \begin{cases} 1 & \text{Quando } N=0 \\ N \times \text{FAT}(N-1) & \text{Quando } N>0 \end{cases}$$

Aula 3 – Funções Recursivas em C

Exemplo

Rodar Programa da função fatorial fat
recorrente (AVA)

Aula 3 – Funções Recursivas em C

A função POT(base,expoente) pode ser escrita na seguinte forma recursiva

$$\text{POT}(\text{base}, \text{expoente}) = \begin{cases} 1 & \text{expoente} = 0 \\ \text{base} \times \text{POT}(\text{base}, \text{expoente} - 1) & \text{expoente} > 0 \end{cases}$$

Aula 3 – Funções Recursivas em C

Exemplo

Rodar Programa da função potência pot
recorrente (AVA)

Aula 4 - Vetores e Strings no C

Um Vetor é uma Estrutura em que uma variável armazena um número predeterminado de valores de mesmo tipo

Ex : $X = (7, -6, 8, 9, 10)$ (Vetor de tamanho 5)

**Componentes : $X[0]=7; X[1]=-6; X[2]=8$
 $X[3]=9; X[4]=10$**

Aula 4 - Vetores e Strings no C

DECLARAÇÃO DE VARIÁVEL VETOR NO C

TIPO NOME_DA_VARIAVEL[TAMANHO];

Ex : int X[5]; (Vetor de Inteiros de Tamanho 5)

**Pode-se no início do programa atribuir valores
(opcional)**

Exemplo : int X[5]={7,-6,8,9,10};

Aula 4 - Vetores e Strings no C

Rodar Programa Vetor em C (AVA)

Rodar programa conversão inteiro decimal para binário (AVA)

Aula 4 - Vetores e Strings no C

CRIAÇÃO DE VARIÁVEIS STRINGS NO C

Não existe tipo string no C;

Para criar uma string utiliza-se um vetor de caracteres

Ex : char palavra[5] (String de tamanho 5)

Aula 4 – Vetores e Strings no C

MANIPULANDO VARIÁVEIS STRINGS NO C

Leitura : `scanf(“%s”,nome_variavel)` (sem &)

Exemplo : `scanf(“%s”,x)`

Observação : A leitura acima só lê string até aparecer o espaço em branco

Isto é se após o `scanf` escrever Rogério Santos será armazenado apenas Rogério (Lê apenas a primeira palavra)

Para ler todas as palavras : `scanf(“%[^\n]”, nome_variavel);`

Exemplo : `scanf(“%[^\n]”,x);`

Observação : `^[^\n]` Expressão regular para ignorar espaço em branco entre palavras

Aula 4 – Vetores e Strings no C

MANIPULANDO VARIÁVEIS STRINGS NO C

As funções a seguir utilizam obrigatoriamente a biblioteca string.h

Atribuição : A string é vetor logo não se pode fazer atribuição

`nome_da_variavel = Valor da String` Ex: Não funcionará `x="Rogerio";`

Logo utiliza-se a função de atribuição :

`strcpy(x, Valor da String)` Exemplo : `strcpy(x, "Rogerio");`

Aula 4 – Vetores e Strings no C

MANIPULANDO VARIÁVEIS STRINGS NO C

As funções a seguir utilizam obrigatoriamente a biblioteca string.h

Comparação entre strings : strcmp (Valor da primeira String ,Valor da Segunda String)

Se der 0 as Strings são iguais, caso contrário serão diferentes

Exemplo : x vale “Rogerio” e y vale “Rogerio” logo strcmp(x,y) dará 0

Comparação dos primeiros n caracteres : strncmp (Valor da primeira String ,Valor da Segunda String,n)

Observação :x vale “Curso C” e y vale “Curso Java” logo strncmp(x,y,5) dará 0

Aula 4 – Vetores e Strings no C

MANIPULANDO VARIÁVEIS STRINGS NO C

As funções a seguir utilizam obrigatoriamente a biblioteca string.h

Concatenação (junção) strings : strcat (Valor da primeira String ,Valor da Segunda String)

O resultado será uma String com os dois valores concatenados

Exemplo : x vale “Rogerio” e y vale “Malheiros” logo strcat(x,y) será RogerioMalheiros

Aula 4 - Vetores e Strings no C

Rodar Programa de Operações com Variável String
(AVA)

Aula 5 - Matrizes no C

Uma Matriz é uma Estrutura em que uma variável armazena valores de mesmo tipo dispostos em M linhas e N colunas (M e N são predeterminados)

Exemplo: $A = \begin{bmatrix} 5 & 8 & 9 \\ 7 & 1 & 3 \end{bmatrix}$ (M=2 e N=3)

$A[0][0]= 5$ $A[0][1]=8$ $A[0][2] =9$ $A[1][0]= 7$ $A[1][1]=1$ $A[1][2] =3$

Aula 5 - Matrizes no C

DECLARAÇÃO DE VARIÁVEL Matriz NO C

TIPO NOME_DA_VARIAVEL[M][N];

Ex : int A[2][3]; (Matriz de inteiros com M=2 N=3)

**Pode-se no início do programa atribuir valores
(opcional)**

Exemplo : int A[2][3]={5,8,9,7,1,3};

Aula 5 - Matrizes no C

Rodar Programa Matrizes em C

Rodar Programa Soma de Matrizes

Rodar Exemplo matriz para leitura e escrita de
medias de turmas(colunas) em unidades(linhas)

(AVA)

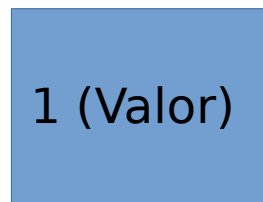
Aula 6 – Ponteiros no C

VARIÁVEIS ESTÁTICAS E DINÂMICAS

Toda variável é um espaço de memória alocado em memória que possui nome (identificador), tipo de valor, valor e endereço de memória

X (identificador)

Ex :



Tipo (Inteiro)

&HC00 (Endereço)

Declaração e Atribuição no C : int X=1;

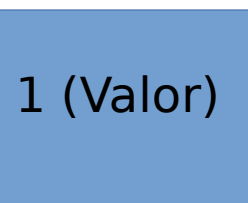
Aula 6 – Ponteiros no C

VARIÁVEL ESTÁTICA

É toda a variável em que o endereço não pode ser alterado pelo programador. Até agora somente foram estudadas variáveis estáticas

X (identificador)

Ex :



Tipo (Inteiro)

&HC00 (Endereço)

Declaração e Atribuição no C :

TIPO NOME=VALOR;

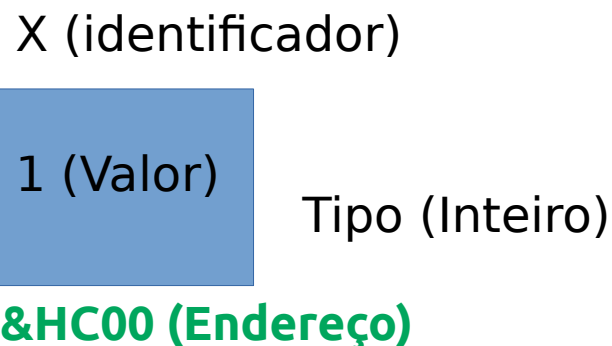
EX: int X=1;

Aula 6 – Ponteiros no C

VARIÁVEL DINÂMICA

É toda a variável em que o endereço pode ser alterado pelo programador.

Ex :



Declaração e Atribuição no C :

TIPO *NOME=VALOR;

EX: int *X=1;

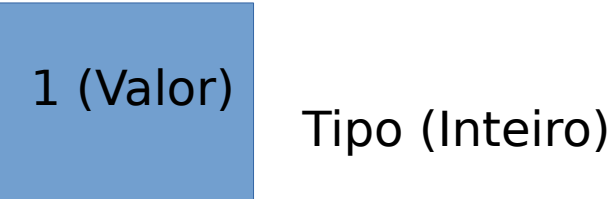
Aula 6 – Ponteiros no C

IMPORTANTE

***NOME se refere ao Valor e NOME ao endereço**

Ex :

X (identificador)



1 (Valor) Tipo (Inteiro)

&HC00 (Endereço)

***X=1; X=&HC00**

Aula 6 – Ponteiros no C

RODAR PROGRAMA DE PONTEIRO NO AVA

1

Aula 7 - Estrutura registro no C (struct)

Considere o problema de fazer o cadastro de Alunos.

Para cada Aluno há nome ,matrícula (código), idade, etc que são de tipos diferentes (dados heterogêneos)

A idéia é guardar estes dados para cada aluno numa variável .

Aula 7 - Estrutura registro no C (struct)

Para isso se cria o tipo struct que :

- serve para agrupar um conjunto de dados heterogêneos (campos) formando um novo tipo de dados**
- Depois de criado o tipo struct antes da função main, dentro do programa declara-se uma variável deste tipo struct .Nesta variável serão guardados os valores dos campos.**

Aula 7 - Estrutura registro no C (struct)

Definição no C (antes da função main())

struct Nome

{

tipo campo1 nome campo1 ;

tipo campo2 nome campo2 ;

. . .

} ;

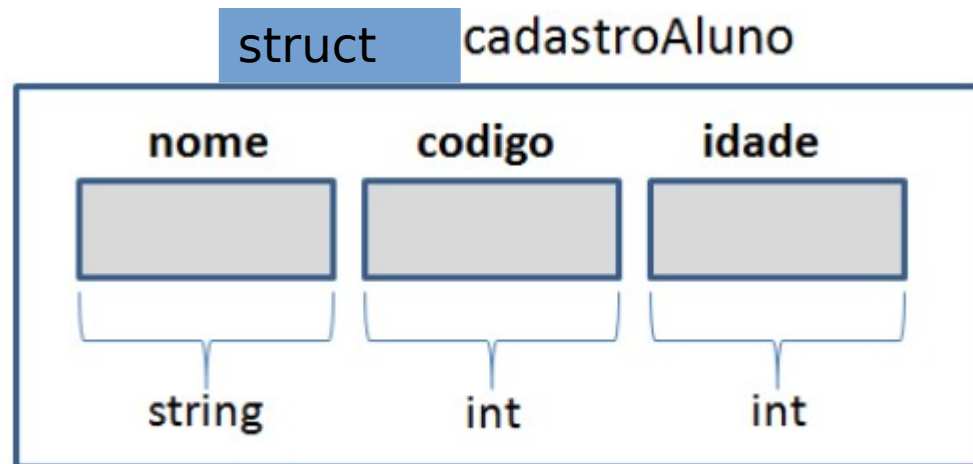
onde :

Nome é como se chamará a estrutura e

tipo_campo1, tipo_campo2, . . . é a lista com os tipos de Dados (campos) em C (char, int, float, double, char[])

Aula 7 - Estrutura registro no C (struct)

Exemplo : Definir o tipo registro de nome cadastroAluno com os dados de Nome , código (matrícula) e idade do Aluno



Aula 7 – Estrutura registro no C (struct)

Exemplo : Definir o tipo registro de nome cadastroAluno com os dados de Nome , código (matrícula) e idade do Aluno no C.

```
struct cadastroAluno  
{  
  c h a r nome [ 5 0 ] ;  
  i n t codigo ;  
  int idade ;  
} ;
```

Aula 7 - Estrutura registro no C (struct)

**Declaração e Atribuição de valores
na Variável do tipo struct**

Declaração

struct nome Identificador;

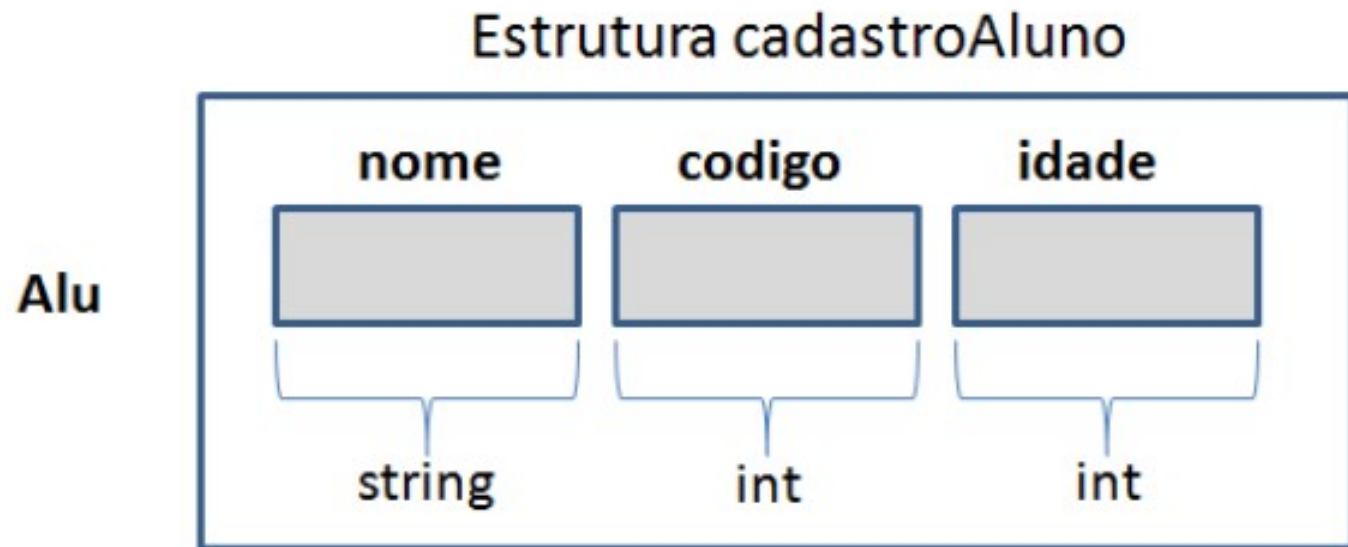
Atribuição de valor ao campo

Identificador.campo=valor;

Aula 7 - Estrutura registro no C (struct)

EXEMPLO :

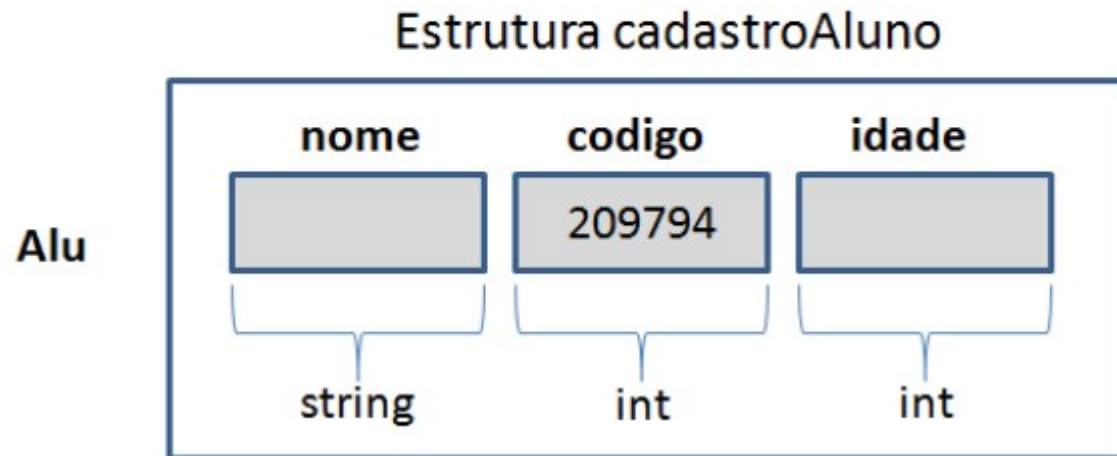
```
struct cadastroAluno Alu;
```



Aula 7 - Estrutura registro no C (struct)

EXEMPLO :

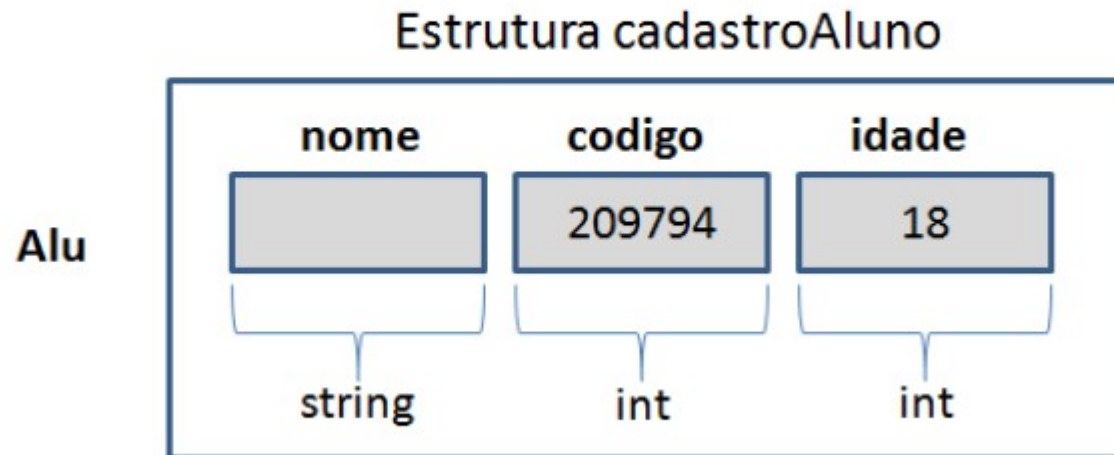
```
Alu.codigo = 209794;
```



Aula 7 - Estrutura registro no C (struct)

EXEMPLO :

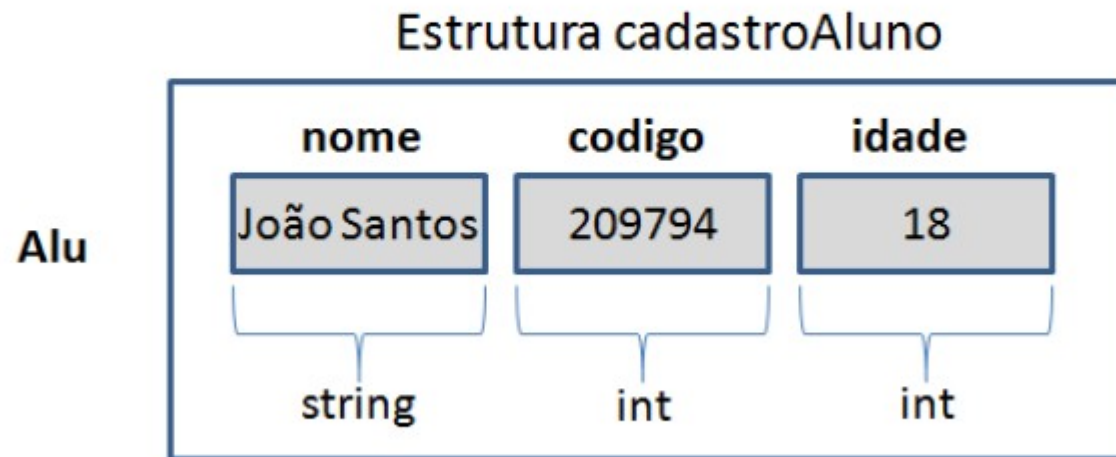
`Alu.idade = 18;`



Aula 7 - Estrutura registro no C (struct)

EXEMPLO :

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



Aula 7 – Estrutura registro no C (struct)

RODAR OS SEGUINTES PROGRAMAS NO AVA

Programa Vetor Registro C

Exemplo Registro Aluno

**Exemplo vetor registros cadastro de
Lista de 20 alunos**