

INSTITUTO FEDERAL
ESPÍRITO SANTO
Campus Cachoeiro de Itapemirim

Registros

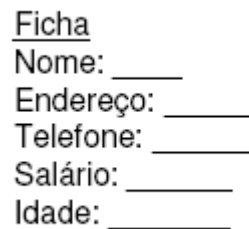
Programação 1

Rafael Vargas Mesquita

<http://www.ci.ifes.edu.br>
<ftp://ftp.ci.ifes.edu.br/informatica/rafael/>

Conceito de Registro (Struct)

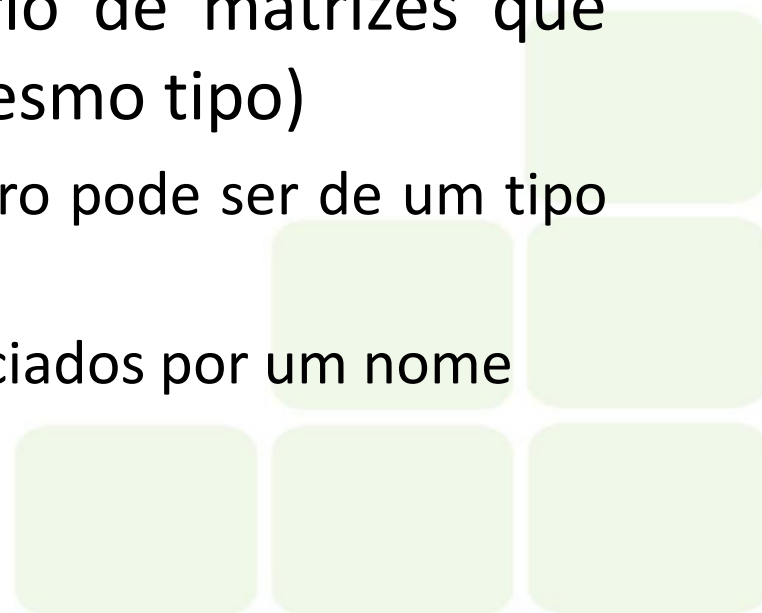
- Vetores e Matrizes
 - Estruturas de dados homogêneas
 - Armazenam vários valores, mas todos de um mesmo tipo (todos int, todos double, todos float, todos char)
- Problemas reais
 - Temos coleções de dados que são de tipos diferentes
 - Exemplo: ficha de um cadastro de cliente
 - Nome: string
 - Endereço: string
 - Telefone: string
 - Salário: float
 - Idade: int



Ficha
Nome: _____
Endereço: _____
Telefone: _____
Salário: _____
Idade: _____

Conceito de Registro (Struct)

- Registro (ou struct)
 - Tipo de dado estruturado heterogêneo
 - Coleção de variáveis referenciadas sobre um mesmo nome
- Permite agrupar dados de diferentes tipos numa mesma estrutura (ao contrário de matrizes que possuem elementos de um mesmo tipo)
 - Cada componente de um registro pode ser de um tipo diferente (int, char, ...)
 - Estes componentes são referenciados por um nome

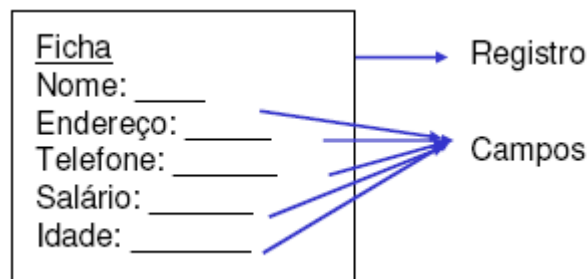


Conceito de Registro (Struct)

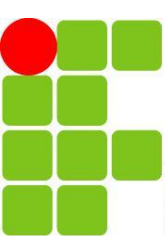
- Os elementos do registro
 - São chamados de campos ou membros da struct
- É utilizado para armazenar informações de um mesmo objeto
- Exemplos:
 - Carro: cor, marca, ano, placa, chassi
 - Pessoa: nome, idade, endereço



Conceito de Registro (Struct)



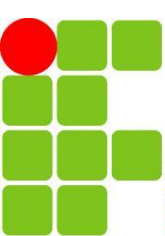
- Campo (Field)
 - Conjunto de caracteres com o mesmo significado
 - Exemplo: nome
- Registro (Struct)
 - Conjunto de campos relacionados
 - Exemplo: nome, endereço, telefone, salário e idade de uma pessoa



Sintaxe na Linguagem C

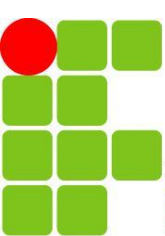
- A palavra reservada struct indica ao compilador que está sendo criada uma estrutura
- Uma estrutura deve ser declarada após incluir as bibliotecas e antes da main

```
struct <identificador_struct> {  
    tipo <nome_variável_campo1>;  
    tipo <nome_variável_campo2>;  
    :  
} <variáveis_estrutura>;  
struct <identificador_struct> <var1>, <var2>;
```



Sintaxe na Linguagem C

- Se o compilador C for compatível com o padrão C ANSI
 - Informação contida em uma struct pode ser atribuída a outra struct do mesmo tipo
 - Não é necessário atribuir os valores de todos os elementos/campos separadamente
 - Por exemplo: `<var1> = <var2>;`
 - Todos os campos de `<var1>` receberão os valores correspondentes dos campos de `<var2>`
- Para acessar os campos da struct
 - Utiliza-se o nome da variável struct, seguido de ponto, seguido do nome do campo
 - Por exemplo: `<var1>.<nome_variável_campo2>;`



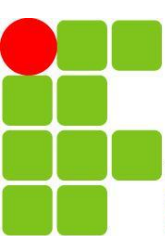
Sintaxe na Linguagem C

- Por exemplo um struct endereço que guarda os elementos nome, rua, cidade, estado e cep

```
struct endereco{  
    char nome[30];  
    char rua[40];  
    char cidade[20];  
    char estado[3];  
    long int cep;  
};
```

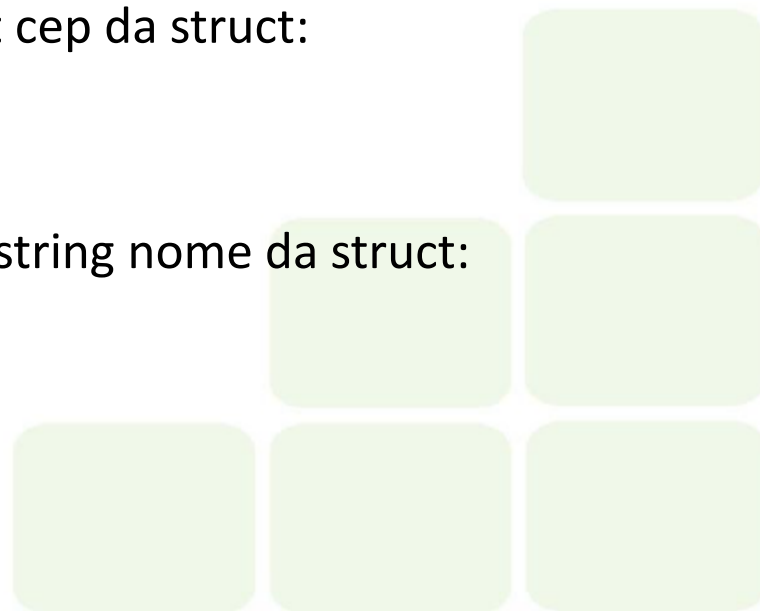
- Foi feita apenas a declaração da struct, ainda não foi criada nenhuma variável da struct endereço
- O comando para declarar uma variável com esta struct é:

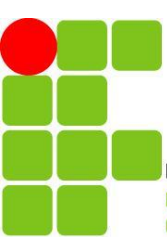
```
struct endereco end;
```

Sintaxe na Linguagem C

- Já vimos que para acessar os membros de uma struct deve-se usar `nome_variável.nome_membro`
- Portanto, considerando o exemplo anterior:
 - Para inicializar o cep da variável `end` que é uma variável da struct endereço:
`end.cep = 123456;`
 - Para obter o cep da pessoa e colocar na int cep da struct:
`scanf ("%d", &end.cep) ;`
 - Para obter o nome da pessoa e colocar na string nome da struct:
`gets (end.nome) ;`
 - Para imprimir a string rua:
`printf ("%s", end.rua) ;`





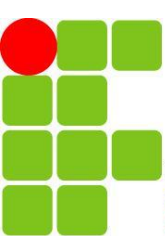
Exemplos

- Exemplo 1:

```
struct aluno {  
    char nome[40];  
    float P1;  
    float P2;  
    int faltas;  
}; //como definição de estrutura é comando, precisa ";"
```

```
main() {  
    struct aluno joao, maria;  
    joao.P1 = 9.5;  
    joao.P2 = 8.5;  
    joao.faltas = 4;  
    maria = joao;  
}
```



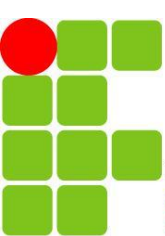


Exemplos

- Exemplo 2:

```
#include <stdio.h>
#include <string.h>
struct endereco {
    char rua[40];
    int num;
    int complemento;
    char cidade[20];
    char estado[3];
    char cep[10];
};
```

```
int main() {
    struct endereco e1; //Na declaração de variáveis do tipo
    struct endereco e2; //"endereco" o compilador aloca
                        // memória para todos os campos
}
```



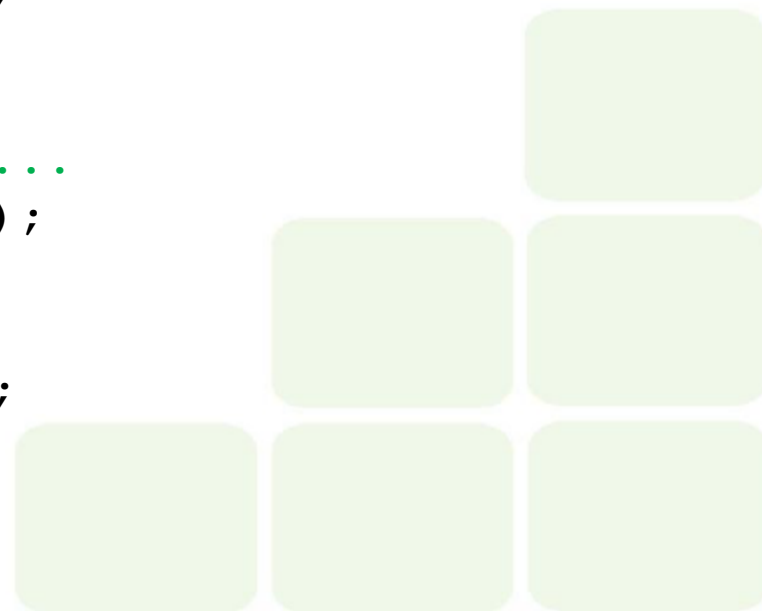
Exemplos

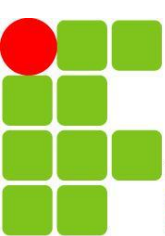
- Exemplo 2:

:

```
// inicialização dos campos de e1...
strcpy(e1.rua, "Avenida Ipiranga");
e1.num = 1234;
e1.complemento = 101;
strcpy(e1.cidade, "Porto Alegre");
strcpy(e1.estado, "RS");
strcpy(e1.cep, "90000-123");
// inicialização dos campos de e2...
strcpy(e2.rua, "Rua Lima e Silva");
e2.num = 1987;
e2.complemento = 308;
strcpy(e2.cidade, "Porto Alegre");
strcpy(e2.estado, "RS");
strcpy(e2.cep, "90000-345");
```

:





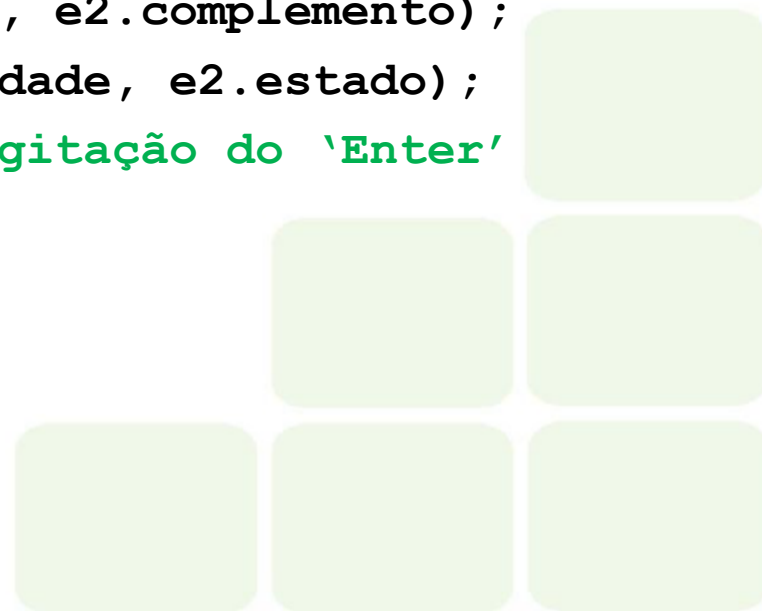
Exemplos

- Exemplo 2:

:

```
// exhibe os dados
```

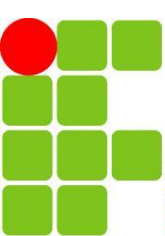
```
printf("\n%s %i/%i", e1.rua, e1.num, e1.complemento);  
printf("\n%s, %s/%s", e1.cep, e1.cidade, e1.estado);  
printf("\n%s %i/%i", e2.rua, e2.num, e2.complemento);  
printf("\n%s, %s/%s", e2.cep, e2.cidade, e2.estado);  
getchar(); //Esta função espera a digitação do 'Enter'  
}
```



Vetor de Registro (Struct)

- O uso mais comum de struct é em vetores
- Para declarar um vetor de struct
 - Define-se a struct
 - Declara-se o vetor do tipo struct criado
- Exemplo:

```
struct aluno vet_alunos[28];  
struct endereco vet_enderecos[100];
```

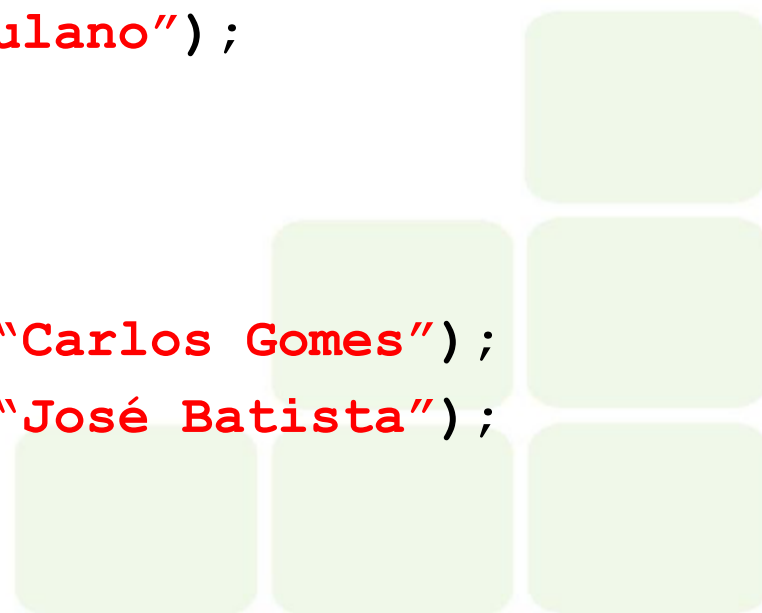


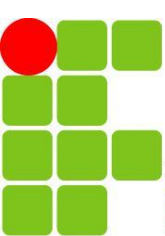
Vetor de Registro (Struct)

- Para manipular os dados do vetor, devem ser fornecidos o índice e o campo

- Exemplo

```
strcpy(vet_alunos[0].nome, "Fulano");  
vet_alunos[0].P1 = 9.5;  
vet_alunos[0].P2 = 8.5;  
vet_alunos[0].faltas = 4;  
strcpy(vet_enderecos[0].rua, "Carlos Gomes");  
strcpy(vet_enderecos[1].rua, "José Batista");
```





Exemplos

- Exemplo:

```
struct endereco{
    char nome[30];
    char rua[40];
    char cidade[20];
    char estado[3];
    long int cep;
};

main()
{
    struct endereco vet_enderecos[100];
    :
    // Imprime todos os nomes do vetor
    for(int i = 0; i < 100; i++)
        printf("%s", vet_enderecos[i].nome);
}
```