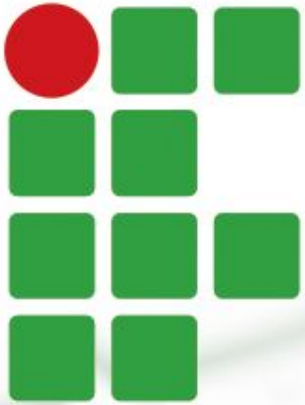


INSTITUTO FEDERAL
Norte de Minas Gerais
Campus Januária

Estruturas de Dados I

- Structs -

Prof. MSc. Adriano Antunes Prates

Structs

- **Imagine a solução para o seguinte problema...**

- Faça um programa que armazene o cadastro de até 100 pessoas.
- Cada cadastro deve armazenar: NOME, CPF, IDADE, ALTURA e PESO.
- Imprima o relatório de todas as pessoas, ordenadas pelo NOME.

SOLUÇÕES?

Structs

- **Imagine a solução para o seguinte problema...**

- Faça um
- Cada cad
- Imprima
pelo NOM

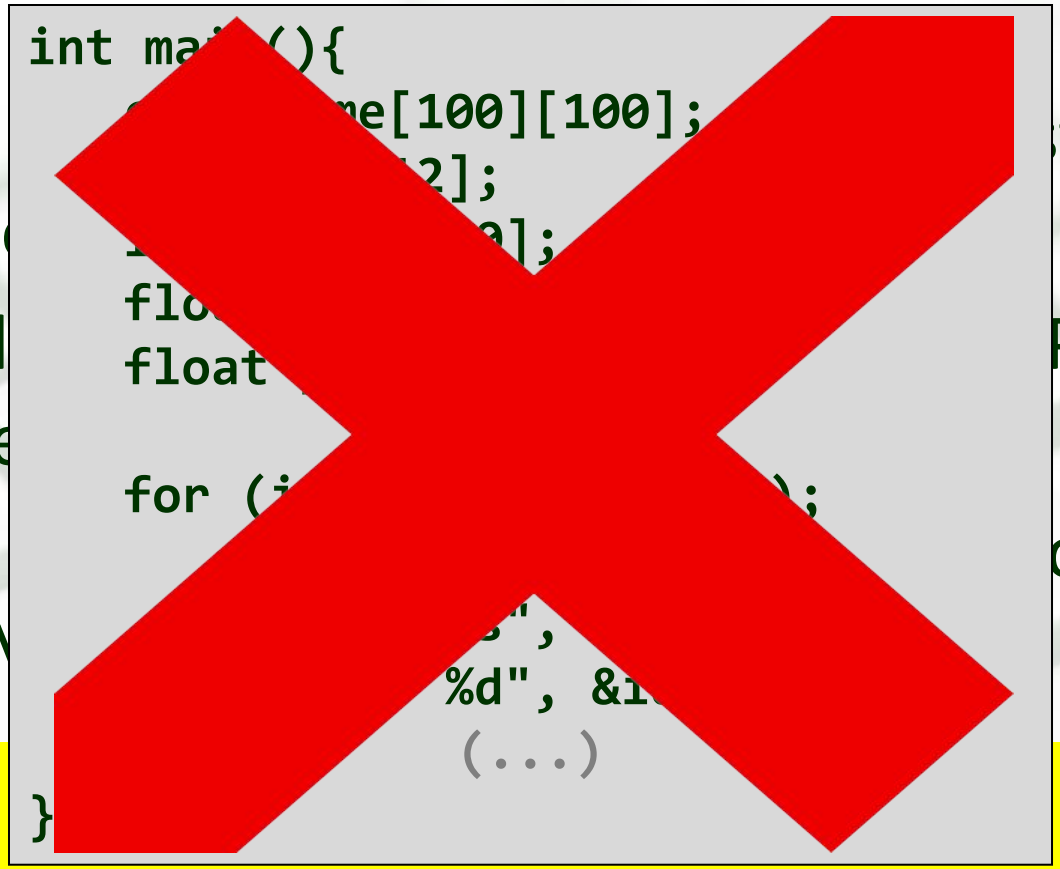
```
int main(){
    char nome[100][100];
    char cpf[12];
    int idade[100];
    float altura[100];
    float peso[100];

    for (int i; i<100; i++);
        scanf(" %s", nome[i]);
        scanf(" %s", cpf[i]);
        scanf(" %d", &idade[i]);
                (...)
}
```

INSTITUTO FEDERAL
Norte de Minas Gerais
Campus Januária

Structs

- **Imagine a solução para o seguinte problema...**
- Faça um
- Cada cad
- Imprima



tro de até
PF, IDADE,
ordenadas

Struct / Registro

- ***STRUCT*** (**registro**) é uma **Estrutura de Dados**:
  - **Composta**: Permite a **agregação** de um conjunto de **valores** sob um mesmo identificador.
  - **Heterogênea**: Estes valores podem ser de um mesmo tipo **ou não**.

- Geralmente, a definição de uma **struct** é feita através da **criação de um novo tipo abstrato de dados (TAD)** com uso do recurso **typedef**.

Declaração typedef

- A declaração **typedef** permite a definição de novos tipos de dados.

```
#include <stdio.h>
                    (...)
typedef int inteiro;
typedef char string[100];
                    (...)
int main(){
    inteiro x,y,z;
    string nome;
}
```

INSTITUTO FEDERAL
Norte de Minas Gerais
Campus Januária

Declaração typedef

- A declaração **typedef** permite novos tipos de dados.

A declaração de um novo tipo de dados tem que ser realizada no escopo global do programa, ou seja, fora de qualquer função ou procedimento.

```
#include <stdio.h>
                    (...)
typedef int inteiro;
typedef char string[100];
                    (...)
int main(){
    inteiro x,y,z;
    string nome;
}
```

Declarando uma Struct

- A declaração de um **novo tipo struct** segue o modelo...

```
typedef struct{
   char nome[100];
   char cpf[12];
   int idade;
   float peso,altura;
}Pessoa;
```

INSTITUTO FEDERAL
Norte de Minas Gerais
Campus Januária

Declarando uma Struct

- A declaração de um **novo tipo struct** segue o modelo...

```
typedef struct{
   char nome[100];
   char cpf[12];
   int idade;
   float peso,altura;
}Pessoa;
```

Declaração de um novo tipo de dados (struct), chamado *"Pessoa"*.

Declarando uma Struct

- A declaração de um **novo tipo struct** segue o modelo...

```
typedef struct{
   char nome[100];
   char cpf[12];
   int idade;
   float peso,altura;
}Pessoa;
```

Variáveis que compõem a estrutura *"Pessoa"*.

Declarando uma Struct

```
#include <stdio.h>

typedef struct{
   char nome[100];
   char cpf[12];
   int idade;
   float peso,altura;
}Pessoa;

int main(){
   Pessoa p;
}
```

Aqui você está declarando uma variável do tipo Pessoa.

INSTITUTO FEDERAL
Norte de Minas Gerais
Campus Januária

ATENÇÃO!

- Imagine que **typedef struct** seja o molde de um carimbo...
  - *Você não escreve em um carimbo em si.*

Mas escreve em um espaço definido por ele.

INSTITUTO FEDERAL
Norte de Minas Gerais
Campus Januária

ATENÇÃO!

TIPO != VARIÁVEL

```c
#include <stdio.h>

int main(){
   int n;
   scanf(" %d", &int);
   return 0;
}
```

Acessando uma Struct

- Observe...

```
int main(){
   Pessoa p;
   scanf(" %[^\n]s",p.nome);
   scanf(" %s",  p.cpf);
   scanf(" %d", &p.idade);
   scanf(" %f", &p.peso);
   scanf(" %f", &p.altura);
}
```

Inicializando Structs

```
Pessoa p = {};
```

```
Pessoa p = {"Adriano", "123", 35, 86, 1.79};
```

```
Pessoa p = {"Adriano", "123"};
```

```
Pessoa p = {35, 86, "Adriano", "123"};
```

```
Pessoa p = {
      .idade=35,
      .peso=86,
      .nome="Adriano",
      .cpf="123"
};
```

Comparação vs. Atribuição

Atribuição de Structs

```
int main(){
   Pessoa a,b;
   scanf(" %[^\n]s", a.nome);
   b = a;
   printf("%s", b.nome);
}
```

Comparação de Structs

```
int main(){
   (...)
   if (a == b)
      printf("A e B são iguais");
}
```

INSTITUTO FEDERAL
Norte de Minas Gerais
Campus Januária

Bora CODAR!!!

1. Faça um programa que define o tipo de dados **Pessoa**, contendo: nome, cpf, altura, peso e idade.

Leia do usuário os dados de uma Pessoa, e após isso:

a) Imprima todas as informações lidas.

b) Calcule o IMC (Índice de Massa Corpórea) dessa pessoa:

IMC = Peso / Altura^2

c) Informe o resultado do IMC, conforme tabela abaixo:

< 18.5	**Abaixo do Peso**
18.5 – 24.9	**Saudável**
25.0 – 29.9	**Acima do Peso**
> 30	**Obesidade**

INSTITUTO FEDERAL
Norte de Minas Gerais
Campus Januária

Trabalhando com N variáveis

- Criar um **novo tipo de dados** para armazenar **apenas 1 variável** não faz muito sentido...
- Normalmente, precisaremos armazenar **muitas variáveis** de um mesmo **tipo struct**...

Qual estrutura de dados permite armazenar vários elementos de um mesmo tipo?

Vetor + Struct == Solução!

```
#include <stdio.h>

typedef struct{
   char nome[100];
   int idade;
   float peso,altura;
   char sexo;
}Pessoa;

int main(){
   Pessoa cadastro[100];
}
```

Declaramos um Array com espaço para 100 elementos do tipo Pessoa

INSTITUTO FEDERAL
Norte de Minas Gerais
Campus Januária

Vetor + Struct == Solução!

```
int main(){
    Pessoa cadastro[100];
}
```





Nome: _____ *CPF: _____ Idade: ____* *Altura: _____ Peso: _____*	*Nome: _____* *CPF: _____ Idade: ____* *Altura: _____ Peso: _____*	*Nome: _____* *CPF: _____ Idade: ____* *Altura: _____ Peso: _____*	*Nome: _____* *CPF: _____ Idade: ____* *Altura: _____ Peso: _____*	*Nome: _____* *CPF: _____ Idade: ____* *Altura: _____ Peso: _____*
0	1	2	...	99

INSTITUTO FEDERAL
Norte de Minas Gerais
Campus Januária

Vetor + Struct == Solução!

```
int main(){
    Pessoa cadastro[100];
}
```





Nome: _____ CPF: _____ Idade: ____ Altura: _____ Peso: _____	*Nome: _____ CPF: _____ Idade: ____ Altura: _____ Peso: _____*	*Nome: _____ CPF: _____ Idade: ____ Altura: _____ Peso: _____*	*Nome: _____ CPF: _____ Idade: ____ Altura: _____ Peso: _____*	*Nome: _____ CPF: _____ Idade: ____ Altura: _____ Peso: _____*
0	1	2	...	99

mas quantas pessoas DE FATO estão cadastradas?

Acessando um Vetor de Struct

```
int main(){
   Pessoa cadastro[100];   //repositório
   int contP = 0;          //qtde itens no repositório
   do{
      scanf(" %[^\n]s",cadastro[contP].nome);
      scanf(" %d", &cadastro[contP].idade);
      scanf(" %f", &cadastro[contP].peso);
      scanf(" %f", &cadastro[contP].altura);
      scanf(" %c", &cadastro[contP].sexo);
      contP++;
      scanf(" %c", &continua)
   }while(continua == 's' && contP<100);
}
```

Acessando um Vetor de Struct

```
int main(){
    Pessoa cadastro[100];   //repositório
    int contP = 0;          //qtde itens no repositório
    do{
        scanf(" %[^\n]s",cadastro[contP].nome);
        scanf(" %d", &cadastro[contP].idade);
        scanf(" %f", &cadastro[contP].peso);
        scanf(" %f", &cadastro[contP].altura);
        scanf(" %c", &cadastro[contP].sexo);
        contP++;
        scanf(" %c", &continua)
    }while(continua == 's' && contP<100);
}
```

INSTITUTO FEDERAL
Norte de Minas Gerais
Campus Januária

Bora CODAR!!!

1. Faça um programa que define um novo tipo de dados chamado **Aluno**. Cada registro de Aluno deve conter: **Nome do Estudante** (s), **Número de Matrícula** (i), **Nome da Disciplina** (s), **Nota Final da Disciplina** (i).
   a) Leia os dados de vários alunos (até o nome informado for "`exit`").
   b) Após a fase de cadastro, pergunte ao usuário do sistema algum **Número de Matrícula** para ser pesquisado, e encontrando o registro, imprima todas as informações deste aluno.
   c) Repita a operação da letra B acima, até que o usuário informe um Nº de matrícula negativo (para indicar a finalização do programa).

2. Refatore o problema anterior. Agora, imagine que um mesmo aluno possa se matricular em (NO MÁXIMO) até 10 disciplinas distintas. Toda disciplina deve possuir uma nota final individualizada. Como modelar a nova *struct* atendendo ao novo requisito? Implemente a solução.

3. Imagine agora que cada aluno possa ter um número variável de disciplinas, (desde 1 até N). Pense (e implemente) uma solução tecnicamente viável.