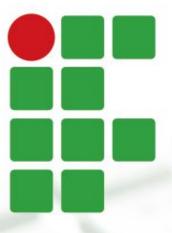
Instituto Federal do Norte de Minas Gerais - IFNMG - Campus Januária Bacharelado em Sistemas de Informação - BSI



INSTITUTO FEDERAL

Norte de Minas Gerais Campus Januária

Estruturas de Dados I

- Structs -



Structs

- Imagine a solução para o seguinte problema...
- Faça um programa que armazene o cadastro de até 100 pessoas.
- Cada cadastro deve armazenar: NOME, IDADE, SEXO, ALTURA e PESO.
- Imprima o relatório de todas as pessoas, ordenadas pelo NOME.

SOLUÇÕES?



Structs

Imagine a solução para o seguinte problema...

- Faça um100 pesso
- Cada cadSEXO, AL
- Imprima pelo NON

```
int main(){
   char nome[100][100];
                                    tro de até
   int idade[100];
   char sexo[100];
   float altura[100];
                                    DADE,
   float peso[100];
   for (int i; i<100; i++);
                                    ordenadas
      scanf(" %s", nome[i]);
      scanf(" %d", idade[i]);
      scanf(" %c", sexo[i]);
               (...)
```



Structs

Imagine a solução para o seguinte problema...

- Faça um100 pesso
- Cada cadSEXO, AL
- Imprima pelo NON





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 <u>ou não</u>;

 Geralmente, a criação de uma struct é feita através da definição 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.



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.



A declaração de um novo tipo <u>struct</u> segue o modelo...

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



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

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

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



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

```
typedef struct{
  char nome[100];
  int idade;
  float peso,altura;
  char sexo;
}Pessoa;
Variáveis que compõem
  a estrutura "Pessoa".
```



```
#include <stdio.h>
typedef struct{
   char nome[100];
   int idade;
   float peso, altura;
   char sexo;
}Pessoa;
int main{
                                Aqui você está declarando
   Pessoa p1;
                               uma variável do tipo Pessoa.
```



ATENÇÃO!

TIPO!= VARIÁVEL

```
#include <stdio.h>
int main{
  int n;
  scanf(" %d", &int);
  return 0;
}
```





Acessando uma Struct

Observe...

```
int main(){
  Pessoa p1;
  scanf(" %[^\n]s",p1.nome);
  scanf(" %d", &p1.idade);
  scanf(" %f", &p1.peso);
  scanf(" %f", &p1.altura);
  scanf(" %c", &p1.sexo);
```



Vamos à Prática!

1. Faça um programa que define um novo tipo de dados, chamado **Pessoa** (nome, altura, peso e idade).

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

- a) Imprima as informações lidas;
- b) Calcule o IMC (Índice de Massa Corpórea) dessa pessoa;

- c) Informe o resultado do IMC:
 - < 18.5 Abaixo do Peso
 - 18.5 24.9 Saudável
 - 25.0 29.9 Acima do Peso
 - > 30 Obesidade



Trabalhando com N variáveis

 Criar um novo tipo de dados para armazenar apenas 1 variável não faz muito sentido...

Normalmente, precisaremos armazenar N
 variáveis de um mesmo tipo struct...

Qual estrutura de dados permite o armazenamento de N elementos de um mesmo tipo de dados?



Vetor + Struct = Solução!

```
#include <stdio.h>
typedef struct{
   char nome[100];
   int idade;
  float peso, altura;
   char sexo;
}Pessoa;
int main{
                                Declaramos um vetor com 100
  Pessoa cadastro[100];
                                  elementos do tipo Pessoa
```



Acessando um Vetor de Struct

Observe...

```
int main(){
  Pessoa cadastro[100];
  for (int i=0; i<100; i++){
     scanf(" %[^\n]s", cadastro[i].nome);
     scanf(" %d", &cadastro[i].idade);
     scanf(" %f", &cadastro[i].peso);
     scanf(" %f", &cadastro[i].altura);
     scanf(" %c", &cadastro[i].sexo);
```



Comparação vs. Atribuição

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

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



Vamos à Prática!

- Faça um programa que define um novo tipo de dados chamado Aluno.
 Cada registro de aluno deve conter os dados: Nome (s), Sexo (c),
 Matricula (i) e Periodo (i).
 - a) Leia 5 registros de aluno.
 - b) Após a leitura, pergunte ao usuário o **nome** de um aluno para ser pesquisado, e imprima todas as informações deste.
 - c) Repita a operação da letra B, até que o usuário informe a palavra "exit".
- 2. Modifique o exemplo anterior, incluindo agora no registro de cada aluno, as notas das 5 disciplinas dele. (*Obs:* As notas devem ser armazenadas em uma estrutura do tipo **vetor**).
- 3. Modifique novamente o exemplo, incluindo agora um **registro** de endereço [rua (s), numero (i), bairro (s)] para cada registro de Aluno. (*Obs:* Endereço deve ser criado como um novo **struct**).