

Variáveis Heterogêneas (Structs)



RECAPITULANDO ...

❑ **Definição informal:**

Conjunto de variáveis integradas, mas todas identificadas e manipuladas a partir de um mesmo nome.

❑ **Variáveis compostas:** podem ser classificadas em dois grandes grupos

1. Homogêneas (Ex.: vetores ou matrizes).

2. Heterogêneas (Ex.: structs, variáveis abstratas, classes, etc).

VARIÁVEIS HETEROGÊNEAS

- Com vetores e matrizes, é possível trabalhar com um número qualquer de variáveis alocadas em sequência, porém, todas essas variáveis **tem que ser de um mesmo tipo**.
- As vezes necessitamos trabalhar com várias variáveis, **mas de tipos diferentes**. Para isso, existem as **variáveis heterogêneas**.
- A partir dessas variáveis, é possível agrupar os mais variados tipos de dados por meio de uma única “super” variável.

VARIÁVEIS HETEROGÊNEAS

□ Variáveis heterogêneas (características)

- São identificadas **por um único nome**;
- São acessadas das formas mais diversas possíveis;
- **Podem conter dados de tipos diferentes.**

Exemplo

“Super” variável do **tipo aluno**, a qual é definida por:

- Nome: (tipo **string**)
- Matrícula: (tipo **int**)
- Coeficiente_rendimento: (tipo **float**)

VARIÁVEIS HETEROGÊNEAS

- Normalmente, a struct é construída **antes** da main (). exemplo →
- A **struct aluno**, pode ser interpretada como um **tipo de dados** (assim como int, float, etc).
- A partir da criação desse novo tipo, podemos utilizá-lo quantas vezes quiser (inclusive criar um **vetor do tipo aluno**, por exemplo).
- Esse tipo **sempre virá** com os componentes nome, matricula e cr.

```
struct aluno
{
    char nome[100];
    int matricula;
    float cr;
};

int main()
{
    struct aluno aluno1;
    return 0;
}
```

VARIÁVEIS HETEROGÊNEAS

- Assim, uma variável do tipo “aluno”, sempre que declarada, será atribuída à ela uma **string**, um **int** e um **float**.
- Em geral, existem várias formas de acessar as componentes da *struct*.
- Vamos adotar a forma mais simples de acesso às componentes de uma *struct*:
 - Utilização do **ponto (.)**
- **Exemplo:** `aluno1.cf = 5.0;`

```
struct aluno
{
    char nome[100];
    int matricula;
    float cr;
};

int main()
{
    struct aluno aluno1;
    return 0;
}
```

EXERCÍCIO 1

1. Crie um programa que: **(a)** crie e preencha uma *struct* do **tipo aluno** (slide anterior) e **(b)** imprima cada um dos campos (components) da struct.
2. Dado **n** (número de alunos), crie um vetor de alunos, preenchendo seus dados em seguida. Imprima o vetor de alunos no final do programa.



**FUI DORMIR ANTES DE
TERMINAR O PROGRAMA**

**AGORA NÃO FAÇO A MÍNIMA IDEIA
DO QUE EU TAVA FAZENDO**

EXERCÍCIO2

- Crie uma struct para representar um número complexo, compostos por uma parte real e uma parte imaginaria.
- Leia dois números complexos z e w , e realize: a soma e o produto entre eles.