

Linguagens de Programação 1

Francisco Sant'Anna
Sala 6020-B

`francisco@ime.uerj.br`

`http://github.com/fsantanna-uerj/LP1`

Tipos Compostos

O que é um tipo?

- “Natureza” ou “Classificação” de um dado
- Número, Texto (string), Booleano
 - Número real, inteiro, inteiro de 1 byte, ...
- `v=100 ; type(v)`
- `v=100.0 ; type(v)`
- `int v = 100;`
- `float v = 100;`

Para que tipos?

- Recusar operações inválidas
- Documentar o código
- Especializar por tamanho
- Desempenho

Tipos “Básicos”

- Numéricos: **char**, **short**, **int**, **float**, *etc.*
- Ponteiros: *tipo**
- Vetores/Arrays: *tipo[n]*
- **int** x;
int xs[10];
int* p = xs;
*(p+3) = 100;

structs

structs

- *Record, Registro, Product Type*
- Construtor de tipos novos
 - construtor **E**
 - *campo1 (de tipo1) E campo2 (de tipo2) E ...*
- Exemplo:
 - Um Personagem é representado por um tipo composto por
`forca(int) E energia(int) E experiencia(int)`

```
#include <stdio.h>

struct Personagem {
    int forca;
    int energia;
    int experiencia;
};

int main (void) {
    struct Personagem p1;
    p1.forca      = 10;
    p1.energia     = 100;
    p1.experiencia = 0;

    struct Personagem p2;
    p2.forca      = 13;
    p2.energia     = 150;
    p2.experiencia = 200;

    printf("> %d %d\n", p1.forca, p2.forca);

    return 0;
}
```


Exercício 7.1

- Crie uma `struct` qualquer.
 - Seja criativo!
- Crie uma função `preenche` que recebe um ponteiro para o novo tipo criado e leia os campos para a variável passada, ex.:
 - ```
struct T t;
preenche (&t) ;
printf ("A=%d, B=%d\n", t.a, t.b) ;
```

# Exercício 7.2

- Crie uma nova `struct` que contenha a `struct` do exercício anterior.
  - Seja criativo!
  - A nova `struct` deve ser definida em separado.
- Crie uma função `preenche2` para preencher valores da nova `struct`.
  - A função deve usar a `preenche` do exercício anterior.

## Exercício 7.3

- Crie uma `struct` para guardar um ponto no espaço bi-dimensional com dois inteiros `x` e `y`.
- Crie uma função para preencher 1 ponto.
- Na `main`, crie um vetor com dez pontos.
- Crie uma função para preencher um vetor de pontos.
- Crie uma função que receba um vetor de pontos e retorne o ponto mais distante de  $(0,0)$ .

# Exercício 7.4

- Um jogo possui 10 personagens, cada um com as seguintes características:
  - Um número que representa a sua “identidade”
  - Um posição (x,y) no espaço bi-dimensional
  - Uma quantidade de pontuação (inicialmente 0)
- Crie uma `struct` para representar um personagem
  - A posição também deve ser uma `struct`
- Crie um vetor com 10 personagens



**unions**

# unions

- União, Variante, *Sum Type*
- Construtor de tipos novos
  - construtor **OU**
  - *campo1 (de tipo1) OU campo2 (de tipo2) OU ...*
- Exemplo:
  - Uma Identidade é representada por um tipo composto por  
`IFP(int) OU CPF(int) OU NOME(char[255])`

```
#include <stdio.h>
#include <string.h>

union Identidade {
 int ifp;
 int cpf;
 char nome[256];
};

int main (void) {
 union Identidade i1;
 i1.ifp = 117766118;

 union Identidade i2;
 i2.cpf = 1688833355;

 union Identidade i3;
 strcpy(i3.nome, "Francisco Sant'Anna");

 printf("> %d %d %s\n", i1.ifp, i2.cpf, i3.nome);

 return 0;
}
```



# União Discriminada

- O que acontece se gravar como um subtipo e ler como outro?
- C possui tipagem fraca
  - Mesmo problema do tamanho de um vetor!
- Solução?
  - Guardar o subtipo em uso

```
struct Identidade {
 int sub;
 union {
 int ifp;
 int cpf;
 char nome[255];
 };
};
```

# Exercício 7.6

- Crie uma `union` qualquer.
  - Seja criativo!
  - Misture com o exercício 7.2
- Use união discriminada.

**Teste**  
**Próxima Segunda 26/11**

**Vale 1.5 na P1**

**Conteúdo deste slide**  
**(struct e union)**