

# *Linguagens de Programação 1*

**Francisco Sant'Anna**  
**Sala 6020-B**

`francisco@ime.uerj.br`

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

**Vetores**  
**Arrays**  
**Strings**

# Vetores

```
int xs[4];
```

```
int ys[4];
```

```
printf("%p %p\n", &xs, &ys);
```

```
printf("%p %p %p %p %p %p\n",  
       xs, xs+0, xs+1, xs+2, xs+3, xs+4);
```

```
*(xs+3) = 10; // xs[3]=10
```

```
printf("%d\n", xs[3]);
```

endereço	id	valor
	xs	
	ys	
	...	

# Exercício 6.1

- Crie uma função `preenche` que receba um vetor `vec` de inteiros, o tamanho do vetor `n` e preencha o vetor com `n` números lidos do teclado.
- Crie uma função `media` que receba um vetor `vec`, o tamanho do vetor `n` e retorne a média entre todos os valores do vetor.
- Dentro das funções, use a notação de ponteiros em vez da de índices (`*` vs `[]`)

# Arrays

- Vetor: array de dimensão 1
  - `int vs[2];`
- C também suporta arrays de múltiplas dimensões
  - `int vs[3][2];`
  - 3 linhas e 2 colunas

# Arrays

```
int vs[3][2] = { {1,2},{3,4},{5,6} };  
printf("%d %d\n", vs[0][1], vs[1][0]);  
printf("%p %p\n", &vs, &vs[0][0]);  
printf("%p\n",      &vs[0][1]);  
printf("%p\n",      &vs[1][0]);
```

endereço	id	valor
	VS	

## Exercício 6.2

- Crie uma função `preenche` que receba um array bidimensional `arr` de inteiros com uma dimensão fixa ( $L \times C$ ), e preencha o array com  $l * c$  números lidos do teclado.
- Crie uma função `media` que receba um array bidimensional `arr`, a quantidade de linhas `l`, a quantidade de colunas `c`, e retorne a média entre todos os valores do array.

## Exercício 6.3

- 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  $(0,0)$ .



# Strings

```
char s1[] = "abc";
```

```
char s2[] = "def";
```

```
printf("%s/%s\n", s1, s2);
```

```
printf("%d\n", strlen(s1));
```

```
printf("%d %d %d %d %d\n",  
      s1[0], s1[1], s1[2], s1[3], s1[4]);
```

endereço	id	valor
	s1	

# Strings

```
char s1[] = "abc";
```

```
char s2[4];
```

```
char s2[0] = 'a';
```

```
char s2[1] = 'b';
```

```
char s2[2] = 'c';
```

```
char s2[3] = '\\0'; // 0
```

```
printf("%s/%s\\n", s1, s2);
```

# Exercício 6.4

- Implementar a função `strlen`:
  - Recebe uma string
  - Retorna a quantidades de caracteres da string
- Implementar a função `strjoin`:
  - Recebe uma string de destino
  - Recebe duas strings de origem
  - Junta em destino as duas strings de origem
  - Deve usar a `strlen`