

# Vetores

Vanessa Braganholo  
vanessa@ic.uff.br

# Exemplo Motivacional

---

- ▶ Programa para auxiliar a escrever “Parabéns!” nas melhores provas de uma disciplina com 3 alunos
  - ▶ Ler os nomes e as notas de 3 alunos
  - ▶ Calcular a média da turma
  - ▶ Listar os alunos que tiveram nota acima da média



# Exemplo Motivacional

---

```
nome1 = input('Informe o nome do aluno 1: ')
nome2 = input('Informe o nome do aluno 2: ')
nome3 = input('Informe o nome do aluno 3: ')

nota1 = float(input('Informe a nota de ' + nome1 + ':'))
nota2 = float(input('Informe a nota de ' + nome2 + ':'))
nota3 = float(input('Informe a nota de ' + nome3 + ':'))

media = (nota1 + nota2 + nota3)/3
print('A media da turma foi', media)

if nota1 > media:
    print('Parabens', nome1)
if nota2 > media:
    print('Parabens', nome2)
if nota3 > media:
    print('Parabens', nome3)
```



# E se fossem 40 alunos?

---

- ▶ É possível definir variáveis que guardam mais de um valor de um mesmo tipo
- ▶ Essas variáveis são conhecidas como variáveis compostas, variáveis subscritas, variáveis indexáveis ou arranjos (*array*)
- ▶ Em Python existem três tipos principais de variáveis compostas:
  - ▶ Listas
  - ▶ Tuplas
  - ▶ Dicionários



# Vetores

---

- ▶ Variável composta **unidimensional**
  - ▶ Contém espaço para armazenar diversos valores
  - ▶ É acessada via um índice
- ▶ A ideia de vetor é comum na matemática, com o nome de variável subscrita
  - ▶ Exemplo:  $x_1, x_2, \dots, x_n$
- ▶ O que vimos até agora são variáveis com somente um valor
  - ▶ Exemplo:  $y = 123$
- ▶ No caso de vetores, uma mesma variável guarda ao mesmo tempo múltiplos valores
  - ▶ Exemplo:  $x_1 = 123, x_2 = 456, \dots$
  - ▶  $x = [123, 456, \dots]$



# Listas

---

- ▶ Em outras linguagens de programação, listas são chamadas de **vetores** e possuem restrições que Python não impõe:
  - ▶ Em Python, os valores de uma lista podem ser de qualquer tipo
  - ▶ Em outras linguagens, os valores precisam ser do mesmo tipo
- ▶ Em Python
  - ▶ `lista = ['A', 1, 2, 'Casa', 2.3]`
  - ▶ `notas = [10, 5, 6.7, 2, 7.5]`

# Utilização de listas

---

- ▶ Para acessar (ler ou escrever) uma posição do vetor, basta informar a posição entre colchetes

```
notas = [8.0, 5.5, 1.5]  
media = (notas[0] + notas[1] + notas[2]) / 3
```

notas	0	8.0
	1	5.5
	2	1.5
media		5.0



# Utilização de listas

---

- ▶ Pode-se iterar por todos os seus valores usando um comando **for**

```
notas = [8.0, 5.5, 1.5]
for i in range(3):
    print(notas[i])
```





# Criação de uma lista a partir de valores lidos do teclado

---

- ▶ Armazenar as notas de 3 alunos em uma lista. A nota de cada aluno será informada pelo teclado.

```
notas[0] = float(input('Digite a nota do primeiro aluno: '))
notas[1] = float(input('Digite a nota do segundo aluno: '))
notas[2] = float(input('Digite a nota do terceiro aluno: '))
```

# Criação de uma lista a partir de valores lidos do teclado

- ▶ Armazenar as notas de 3 alunos em uma lista. A nota de cada aluno será informada pelo teclado.

```
notas[0] = float(input('Digite a nota do primeiro aluno: '))
notas[1] = float(input('Digite a nota do segundo aluno: '))
notas[2] = float(input('Digite a nota do terceiro aluno: '))
```

Digite a nota do primeiro aluno: 8

Traceback (most recent call last):

File "/Users/vanessa/workspace/PyCharmProjects/AloMundo/notas.py",  
line 1, in <module>

notas[0] = eval(input('Digite a nota do primeiro aluno: '))

**NameError: name 'notas' is not defined**

Process finished with exit code 1

## É preciso primeiro criar a lista...

---

- ▶ Como não sabemos o que colocar em cada posição da lista, vamos criar uma lista vazia

```
notas = []
```

- ▶ Depois vamos adicionar valores na lista usando **append**

```
n = float(input('Digite a nota do primeiro aluno: '))  
notas.append(n)
```

# Voltando ao exemplo

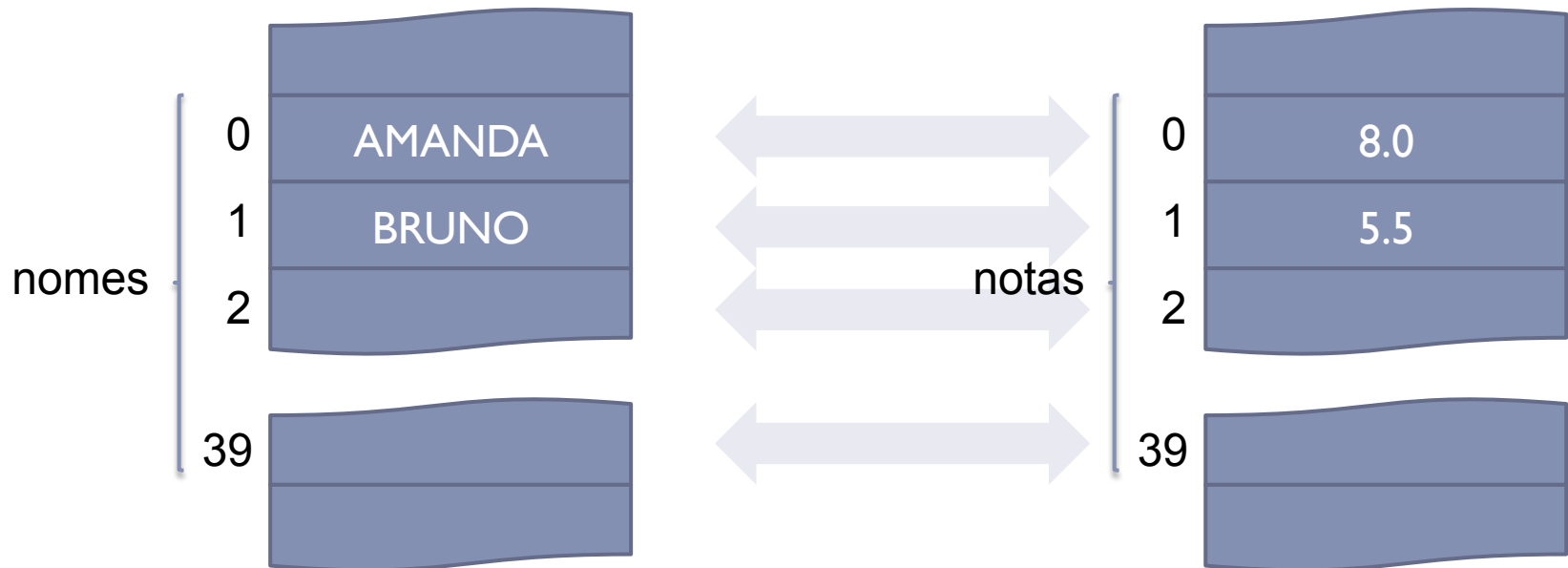
---

- ▶ Armazenar as notas de 3 alunos em uma lista. A nota de cada aluno será informada pelo teclado.

```
notas = []
notas.append(float(input('Digite a nota do primeiro aluno: ')))
notas.append(float(input('Digite a nota do segundo aluno: ')))
notas.append(float(input('Digite a nota do terceiro aluno: ')))
print(notas)
```

## Retomando: E se fossem 40 alunos?

- ▶ Criaríamos dois vetores (nomes e notas) de 40 posições
- ▶ Vincularíamos a posição  $N$  do vetor de nomes à posição  $N$  do vetor de notas



# Retomando: E se fossem 40 alunos?

---

```
num_alunos = 40
nomes = []
notas = []
media = 0

for i in range(num_alunos):
    nomes.append(input('Informe o nome do aluno: '))
    notas.append(float(input('Informe a nota de ' + nomes[i] + ': ')))
    media = media + notas[i]

media = media / num_alunos
print('A media da turma eh ', media)

for i in range(num_alunos):
    if notas[i] > media:
        print('Parabens', nomes[i])
```



# Cuidados no uso de listas

---

- ▶ Certifique-se de que não esteja querendo acessar posição da lista que não existe
- ▶ Exemplo:

```
alunos = ['Andre', 'Lucas', 'Antonio', 'Maria']  
print(alunos[4])
```

# Cuidados no uso de listas

---

- ▶ Certifique-se de que não esteja querendo acessar posição da lista que não existe
- ▶ Exemplo:

```
alunos = ['Andre', 'Lucas', 'Antonio', 'Maria']  
print(alunos[4])
```

```
Traceback (most recent call last):  
  File "/Users/vanessa/workspace/PyCharmProjects/AloMundo/notas.py",  
    line 2, in <module>  
      print(alunos[4])  
IndexError: list index out of range
```

```
Process finished with exit code 1
```



# Índices para acesso aos elementos da lista

---

- ▶ Python permite acesso à lista em ordem crescente ou decrescente de posição
  - ▶ Primeira posição é 0
  - ▶ Última posição é -1

```
>>> c = [-45, 6, 0, 72, 1543]
>>> c[3]
72
>>> c[-2]
72
>>> c[0] = c[-5]
True
```

c[0]	-45	c[-5]
c[1]	6	c[-4]
c[2]	0	c[-3]
c[3]	72	c[-2]
c[4]	1543	c[-1]

# Funções de manipulação de listas

---

- ▶ **len(lista)**

- ▶ Retorna o tamanho da lista

```
>>> numeros = [3, 1, 6, 7, 10, 22, 4]
```

```
>>> len(numeros)
```

```
7
```

# Exemplo

---

- ▶ Programa que lê uma lista do teclado, soma 1 aos elementos da lista e imprime a lista resultante

```
continua = 's'
lista = []
while (continua == 's' or continua == 'S'):
    n = int(input('Digite um numero: '))
    lista.append(n)
    continua = input('Deseja continuar? (s/n): ')

print(lista)

for i in range(len(lista)):
    lista[i] = lista[i] + 1

print(lista)
```

# Concatenação de listas

---

- ▶ É possível anexar os valores de uma lista em outra usando o operador “+”

```
>>> lista = [1,2,3]
>>> lista = lista + [4]
[1,2,3,4]
>>> lista = lista + [4,5,6]
[1,2,3,4,4,5,6]
```

# Exemplo

---

- ▶ Programa que retorna uma lista com todos os números pares entre 2 e um número n, inclusive

```
n = int(input('Digite um numero: '))
lista = []
for i in range(2,n+1,2):
    lista = lista + [i]
print(lista)
```

# Exemplo

---

- ▶ Programa que retorna uma lista com todos os números pares entre 2 e um número n, inclusive, **em ordem reversa**

```
n = int(input('Digite um numero: '))
lista = []
for i in range(2,n+1,2):
    lista = [i] + lista
print(lista)
```

# “Multiplicação” de listas

---

- ▶ O operador “\*” repete **n** vezes os elementos que já estão na lista
- ▶ **lista \* n** equivale a **lista + lista + ... + lista** (n vezes)

```
>>> lista = [1,2,3]
>>> lista = lista * 3
[1,2,3,1,2,3,1,2,3]
```

# Inicialização de listas com zero

---

- ▶ Em diversas situações onde já sabemos de antemão qual será o tamanho da lista, é útil inicializar a lista com o valor 0. Isso evita que precisemos usar o append para adicionar valores

```
>>> tamanho = 10
>>> lista = [0] * tamanho
>>> lista
[0, 0, 0, 0, 0, 0, 0, 0, 0, 0]
```



# Exemplo

---

```
# inicializa vetor de notas com 0
notas = [0] * 3
soma = 0
# preenche vetor de notas, sem usar append
for i in range(3):
    notas[i] = float(input("Digite a nota
do aluno " + str(i) + ": "))
    soma = soma + notas[i]
print("A media da turma é", soma/3)
```

# Representação de Listas em Memória

---

- ▶ O valor de uma variável de lista na verdade é um endereço de memória

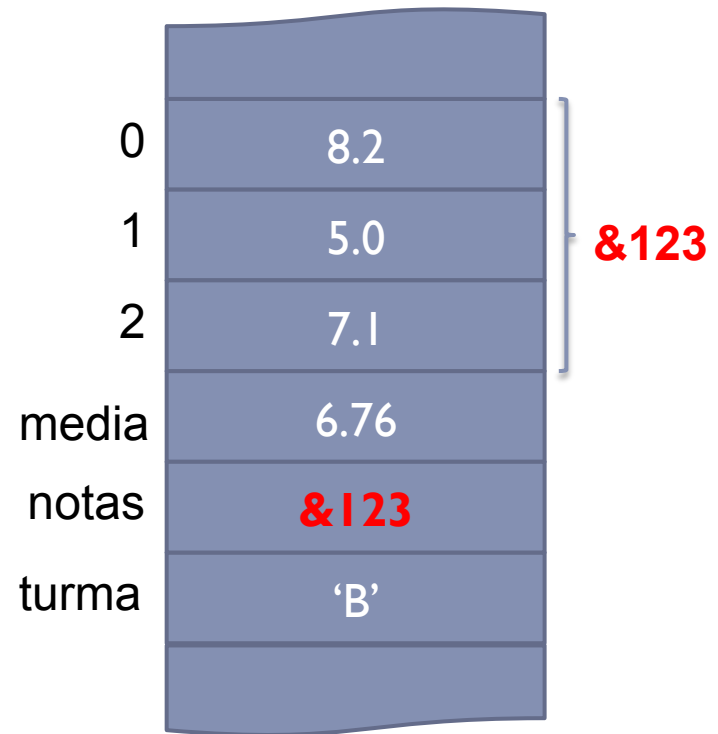


# Representação de Listas em Memória

## Em Python

```
notas = [8.2, 5.0, 7.1]
turma = 'B'
media = 0
for i in range(len(notas)):
    media = media + notas[i]
media = media/len(notas)
```

## Na Memória



# Cópia de listas

---

- ▶ Ao copiar uma lista para outra, o que é feito é copiar o valor do endereço de memória
  - ▶ Ambas passam a apontar para o mesmo endereço, portanto o que for modificado em uma lista também será modificado na outra



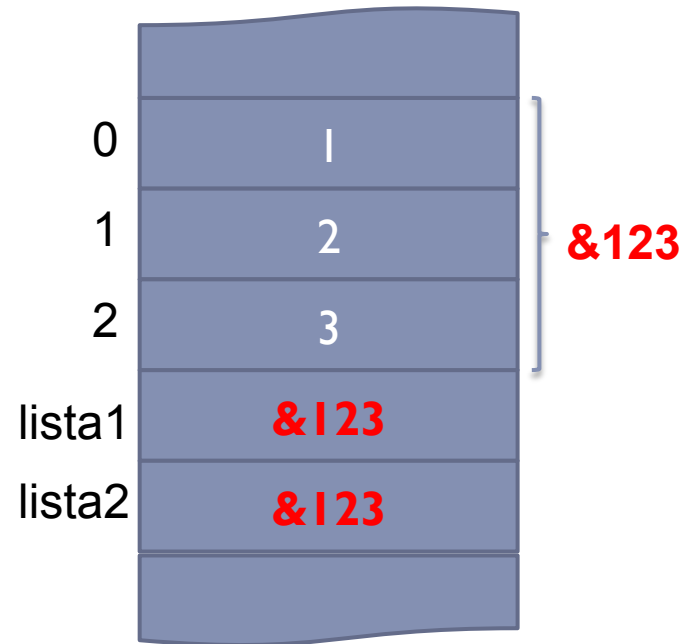
# Cópia de Listas

---

## Em Python

```
>>> lista1 = [1, 2, 3]
>>> lista2 = lista1
```

## Na Memória



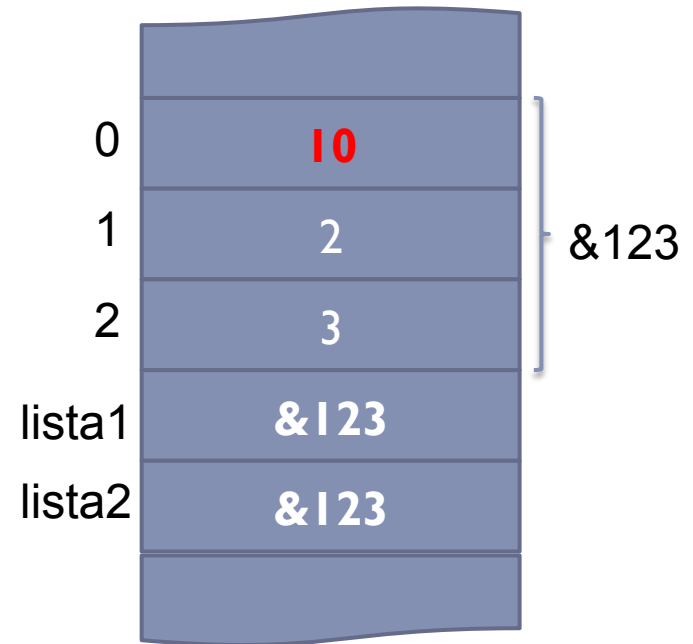
# Cópia de Listas

---

## Em Python

```
>>> lista1 = [1, 2, 3]
>>> lista2 = lista1
>>> lista1[0] = 10
>>> lista1
[10, 2, 3]
>>> lista2
[10, 2, 3]
```

## Na Memória

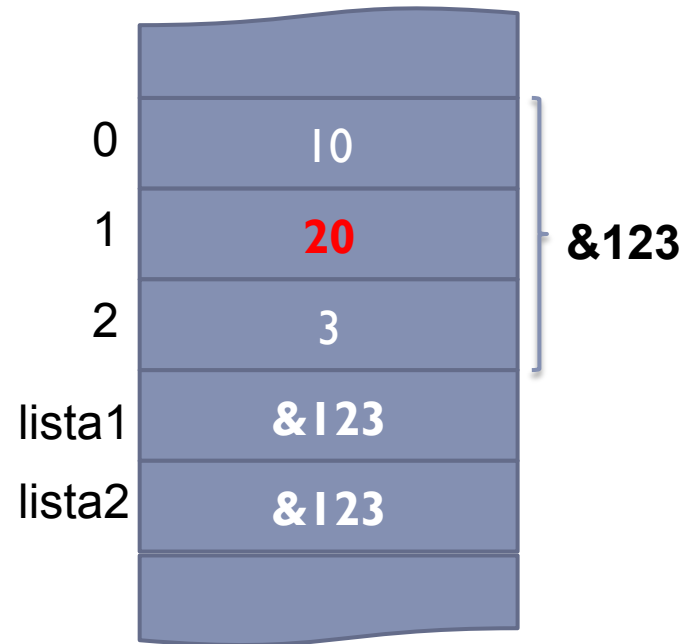


# Cópia de Listas

## Em Python

```
>>> lista1 = [1, 2, 3]
>>> lista2 = lista1
>>> lista1[0] = 10
>>> lista1
[10, 2, 3]
>>> lista2
[10, 2, 3]
>>> lista2[1] = 20
>>> lista2
[10, 20, 3]
>>> lista1
[10, 20, 3]
```

## Na Memória



# Como evitar isso?

---

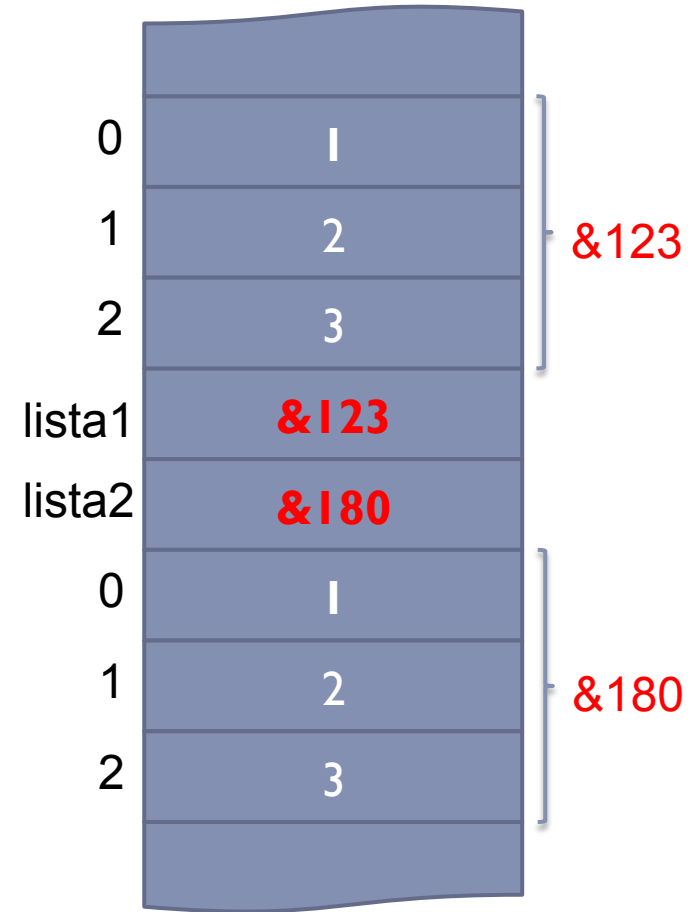
- ▶ Usar um for para copiar valor a valor





# Exemplo

```
>>> lista1 = [1, 2, 3]
>>> lista2 = []
>>> for i in range(len(lista1)):
...     lista2.append(lista1[i])
...
```



## Dessa forma...

---

- ▶ Alterações em uma lista não são refletidas na outra

```
>>> lista1 = [1, 2, 3, 4, 5]
>>> for i in range(len(lista1)):
...     lista2.append(lista1[i])
>>> lista2[0] = 10
>>> lista1
[1, 2, 3, 4, 5]
>>> lista2
[10, 2, 3, 4, 5]
>>> lista1[3] = 20
>>> lista2
[10, 2, 3, 4, 5]
```

---



# Voltando para subprogramação...

---

# Tipos de passagem de Parâmetro

---

- ▶ **Por valor:** o valor da variável na chamada é copiado para a variável da função.
  - ▶ Alterações não são refletidas na variável original
- ▶ **Por referência:** é como se o mesmo “escaninho” fosse usado.
  - ▶ Alterações são refletidas na variável original



# Tipos de passagem de Parâmetro

- ▶ **Por valor:** o valor da variável na chamada é copiado para a variável da função.
  - ▶ Alterações não são refletidas na variável original

Python usa passagem de parâmetro por valor



# Uso de vetores como parâmetros de funções

---


- ▶ Python usa passagem de parâmetro por valor
  - ▶ Faz cópia do valor da variável original para o parâmetro da função
  - ▶ Variável original fica preservada das alterações feitas dentro da função
- ▶ Exceção: **vetores** (ou objetos) funcionam de forma diferente, pois o que é copiado é o **endereço** do vetor, e portanto qualquer alteração é refletida no programa principal → passagem de parâmetro por referência

# Exemplo

---

```
def maior(vetor):  
    #ordena o vetor usando Bubble Sort  
    if len(vetor) > 1:  
        for j in range(0, len(vetor)):  
            for i in range(0, len(vetor)-1):  
                if vetor[i]>vetor[i+1]:  
                    aux = vetor[i+1]  
                    vetor[i+1] = vetor[i]  
                    vetor[i] = aux  
    #retorna o último elemento do vetor  
    return vetor[len(vetor)-1]
```

```
v = [5, 4, 3, 2, 1]  
print(v)  
m = maior(v)  
print(m)  
print(v)
```



O que será  
impresso na tela?

# Exercícios (usar funções sempre que possível)

---

1. Faça um programa que leia dois vetores de 3 posições, que representam forças sobre um ponto no espaço 3D, e escreva a força resultante
  - ▶ Dica: força resultante é obtida pela soma dos valores das coordenadas correspondentes nos dois vetores:  $(x1 + x2)$ ,  $(y1 + y2)$ ,  $(z1 + z2)$
2. Faça um programa que preencha por leitura um vetor de 10 posições, e conta quantos valores diferentes existem no vetor.
3. Faça um programa que preencha por leitura um vetor de 5 posições, e informe a posição em que um valor  $x$  (lido do teclado) está no vetor. Caso o valor  $x$  não seja encontrado, o programa deve imprimir o valor -1





## Exercícios (usar funções sempre que possível)

---

4. Um dado é lançado 50 vezes, e o valor correspondente é armazenado em um vetor. Faça um programa que determine o percentual de ocorrências de face 6 do dado dentre esses 50 lançamentos.
5. Faça um programa que leia um vetor **vet** de 20 posições. O programa deve gerar, a partir do vetor lido, um outro vetor **pos** que contenha apenas os valores inteiros positivos de **vet**. A partir do vetor **pos**, deve ser gerado um outro vetor **semdup** que contenha apenas uma ocorrência de cada valor de **pos**.

# Exercícios (usar funções sempre que possível)

---

6. Leia um vetor de 10 posições e ordene o vetor, usando 3 métodos de ordenação diferentes (crie um programa para cada um)
  - a. Insertion Sort
  - b. Selection Sort
  - c. Bubble Sort
- ▶ Em cada alternativa, conte o número de comparações realizadas, e imprima o número de comparações junto com o vetor ordenado
- ▶ Observe qual dos algoritmos executou a ordenação com o menor número de comparações

# Referências

---

- ▶ Slides baseados nas aulas de Leonardo Murta e Aline Paes

# Vetores

Vanessa Braganholo  
vanessa@ic.uff.br