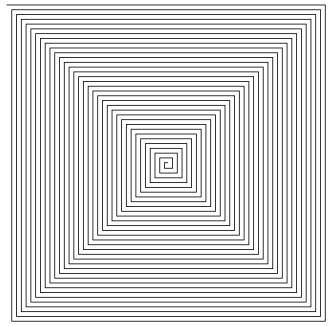
DEPARTAMENTO DE INFORMÁTICA PUC-RIO

Recursão



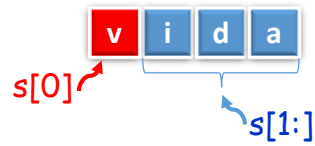
DEPARTAMENTO DE INFORMÁTICA PUC-RIO

Interpretação Recursiva de uma Cadeia de Caracteres (String)

Interpretação recursiva

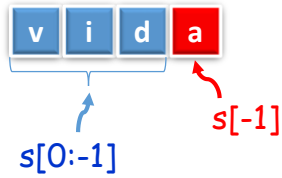
- ✓ Uma cadeia de caracteres pode ser
  - vazia

    OU
  - um caractere seguido de uma cadeia de caracteres



OU

  - um caractere precedido por uma sequência de caracteres



3

Exemplo

- ✓ Crie uma função recursiva chamada ***contaMai*** que retorne a quantidade de letras maiúsculas de uma string ***s*** recebida como parâmetro
- ✓ Exemplo
  - 'Fique EM caSa!' ➔ 4
- ✓ Caso base
  - string vazia ➔ retorna 0
- ✓ Passo recursivo
  - se 1º caractere for letra maiúscula:

    retorna 1 + contaMai(string restante)
  - senão:

    retorna contaMai(string restante)

4

Exemplo

- ✓ Algoritmo

```
# testa caso base
se s == '':                    # se len(s) == 0:
    retorna 0
# testa se 1º caractere é letra maiúscula
se s[0] for letra maiúscula:
    retorna 1 + contaMai(s[1:])
retorna contaMai(s[1:])
```

5

Função *contaMai*

```
def contaMai (s):
    if s == '':
      return 0
    if s[0] >= 'A' and s[0] <= 'Z':
      return 1 + contaMai(s[1:])
    return contaMai(s[1:])

# Chamada da  função
qtdMai = contaMai ('ViDa')
print ('Qtd. de letras maisúculas = %d' %qtdMai)
```

6

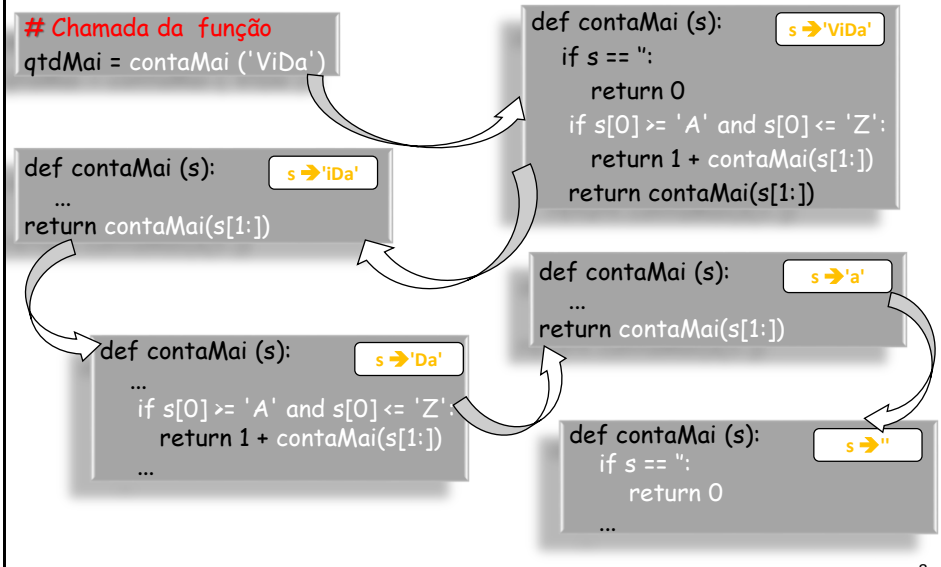
Função *contaMai* – outra versão

```
def contaMai (s):
    if s == '':
      return 0
    qtdMai = contaMai(s[1:])
    if s[0] >= 'A' and s[0] <= 'Z':
      qtdMai = qtdMai + 1
    return qtdMai

# Chamada da  função
qtdMai = contaMai ('ViDa')
print ('Qtd. de letras maisúculas = %d' %qtdMai)
```

7

Simulação da execução



8

Simulação da execução

```
# Chamada da função
qtdMai = contaMai ('ViDa')
```

```
def contaMai (s):
    ...
return contaMai(s[1:])
```
s → 'iDa'

```
def contaMai (s):
    ...
    if s[0] >= 'A' and s[0] <= 'Z':
        return 1 + contaMai(s[1:])
    ...
```
s → 'Da'

```
def contaMai (s):
    if s == '':
        return 0
    if s[0] >= 'A' and s[0] <= 'Z':
        return 1 + contaMai(s[1:])
    return contaMai(s[1:])
```
s → 'ViDa'

```
def contaMai (s):
    ...
return contaMai(s[1:])
```
s → 'a'

0

```
def contaMai (s):
    if s == '':
        return 0
    ...
```
s → ''

Simulação da execução


```
# Chamada da função
qtdMai = contaMai ('ViDa')
```

```
def contaMai (s):
    ...
return contaMai(s[1:])
```
s → 'iDa'

```
def contaMai (s):
    ...
    if s[0] >= 'A' and s[0] <= 'Z':
        return 1 + contaMai(s[1:])
    ...
```
s → 'Da'

```
def contaMai (s):
    if s == '':
        return 0
    if s[0] >= 'A' and s[0] <= 'Z':
        return 1 + contaMai(s[1:])
    return contaMai(s[1:])
```
s → 'ViDa'

```
def contaMai (s):
    ...
return contaMai(s[1:])
```
s → 'a'

0

0

Simulação da execução

```
# Chamada da função
qtdMai = contaMai ('ViDa')
```

```
def contaMai (s):
    ...
return contaMai(s[1:])
```
s → 'iDa'

```
def contaMai (s):
    ...
    if s[0] >= 'A' and s[0] <= 'Z':
        return 1 + contaMai(s[1:])
    ...
```
s → 'Da'

```
def contaMai (s):
    if s == '':
        return 0
    if s[0] >= 'A' and s[0] <= 'Z':
        return 1 + contaMai(s[1:])
    return contaMai(s[1:])
```
s → 'ViDa'

1

0

Simulação da execução

```
# Chamada da função
qtdMai = contaMai ('ViDa')
```

```
def contaMai (s):
    ...
return contaMai(s[1:])
```
s → 'iDa'

1

```
def contaMai (s):
    if s == '':
        return 0
    if s[0] >= 'A' and s[0] <= 'Z':
        return 1 + contaMai(s[1:])
    return contaMai(s[1:])
```
s → 'ViDa'

1

Simulação da execução

```
# Chamada da  função
qtdMai = contaMai ('ViDa')
```

2

```
def contaMai (s):                    s ➔ 'ViDa'
    if s == '':
        return 0
    if s[0] >= 'A' and s[0] <= 'Z':
        return 1 + contaMai(s[1:])   1
    return contaMai(s[1:])
```


Simulação da execução

```
# Chamada da  função
qtdMai = contaMai ('ViDa')
```

2

✓ Na variável qtdMai, será armazenado o valor 2

Exercícios

1. Crie uma função chamada ***trocachar*** que receba uma string e retorne uma nova string substituindo todas as ocorrências de 'A' na string recebida pelo caractere '@'
2. Altere a função acima para que troque as ocorrências de um caractere por outro, ambos recebidos como parâmetro

Exercício

```
def trocachar(s):
  # Testa caso base
  if len(s) == 0:
     return ''
  # chama recursiv. p/string sem o 1° caracter
  novaStr = trocachar(s[1:])
  # acrescenta 1° caracter na nova string
  if s[0]=='A':
     return '@'+ novaStr
  else:
     return s[0] + novaStr
```