



# Introdução à Programação

*Eduardo Silva Lira*

*XLVIII Programa de Verão do IME-USP*

*São Paulo - SP, Jan 2019*



INSTITUTO DE MATEMÁTICA E ESTATÍSTICA  
UNIVERSIDADE DE SÃO PAULO



# Vamos pensar...

- Lembram como armazenávamos um caractere?

# Vamos pensar...

- Lembram como armazenávamos um caractere?
- Como faríamos para trabalhar com textos?

# **Vamos pensar...**

- Lembram como armazenávamos um caractere?
- Como faríamos para trabalhar com textos?

**Vamos direto ao exemplo!**

# Textos em ANSI C

Exemplo:

```
#include<stdio.h>

int main(){
    char msg[50] = "Agora esta ficando interessante!";

    msg[0] = '@';

    printf("A mensagem eh: %s\n", msg);

    printf("Bytes da memoria ocupados: %ld\n", sizeof(msg));

    return 0;
}
```

# Textos em ANSI C

- Um conjunto de caracteres
  - Uma variável com várias posições
  - É um **vetor**!
  - Formatador **%s**

# Textos em ANSI C

- É um **vetor**!
  - Como assim?!

```
char nome[10] = "Amanda";
```



Esta é a variável  
**nome** na memória

# Textos em ANSI C

- É um **vetor**!
  - Como assim?!
  - O que é aquele **\0**?

**char nome[10] = "Amanda";**

'A'	'm'	'a'	'n'	'd'	'a'	'\0'			
-----	-----	-----	-----	-----	-----	------	--	--	--



# Textos em ANSI C

- É um **vetor**!
  - Como assim?!
  - O que é aquele **\0**?
  - Temos **posições**!

**char nome[10] = "Amanda";**

0	1	2	3	4	5	6	7	8	9
'A'	'm'	'a'	'n'	'd'	'a'	'\0'			

# Entrada de Textos - como?

- Com **scanf**

- **COM CUIDADO!**

- Com **fgets**

- **Sim!**

**fgets** (<variavel>, <tamanho>, <origem>);

# Entrada de Textos - como?

- Com **scanf**
  - **COM CUIDADO!**
- Com **fgets**
  - **Sim!**

**fgets** (**nome**, **20**, **stdin**);

# Entrada de Textos - como?

- Com **scanf**

**COM CUIDADO!**

- Com **fgets**

- **Sim!**

**fgets** (**nome**, **20**, **stdin**);

Fonte dos dados:  
Standard Input  
Entrada padrão

Endereço da variável  
onde os dados serão  
armazenados

Quantidade de  
caracteres retornados  
para a variável.  
Inclui o caractere de  
fim (caractere nulo)

# Texto em memória

- Se o usuário digitar “**Perdido!**”, como ficará em memória?

0	1	2	3	4	5	6	7	8	9
'P'	'e'	'r'	'd'	'i'	'd'	'o'	'\0'		

# Texto em memória

- Se o usuário digitar “**Perdido!**”, como ficará em memória?
  - E se lermos novamente e o usuário digitar “**Massa**”?

0	1	2	3	4	5	6	7	8	9
'P'	'e'	'r'	'd'	'i'	'd'	'o'	'\0'		

# Texto em memória

- Se o usuário digitar “**Perdido!**”, como ficará em memória?
  - E se lermos novamente e o usuário digitar “**Massa**”?

0	1	2	3	4	5	6	7	8	9
'M'	'a'	's'	's'	'a'	'\0'	'o'	'\0'		

# Texto em memória

- Se o usuário digitar “**Perdido!**”, como ficará em memória?
  - E se lermos novamente e o usuário digitar “**Massa**”?
    - O **'\0'** sempre marcará o final do texto!

0	1	2	3	4	5	6	7	8	9
'M'	'a'	's'	's'	'a'	'\0'	'o'	'\0'		



# Exemplo

- Crie um programa para contar o tamanho real de um texto
  - Quantidade de caracteres

# Tamanho de um texto

- No começo o tamanho é:

0	1	2	3	4	5	6	7	8	9
'B'	'o'	'm'	' '	'd'	'i'	'a'	'\0'		

# Tamanho de um texto

- No começo o tamanho é: **zero**

**tam = 0**

0	1	2	3	4	5	6	7	8	9
'B'	'o'	'm'	' '	'd'	'i'	'a'	'\0'		

# Tamanho de um texto

- No começo o tamanho é: **zero**

**tam = 0**

Percorra o seu texto começando da posição inicial enquanto não encontrar o '**\0**'.

Cada novo caractere identifica uma letra (some 1 a cada posição percorrida).

0	1	2	3	4	5	6	7	8	9
'B'	'o'	'm'	' '	'd'	'i'	'a'	'\0'		


# Tamanho de um texto

- No começo o tamanho é: zero

**tam = 1**

Percorra o seu texto começando da posição inicial enquanto não encontrar o '`\0`'.

Cada novo caractere identifica uma letra (some 1 a cada posição percorrida).



0	1	2	3	4	5	6	7	8	9
'B'	'o'	'm'	' '	'd'	'i'	'a'	'\0'		

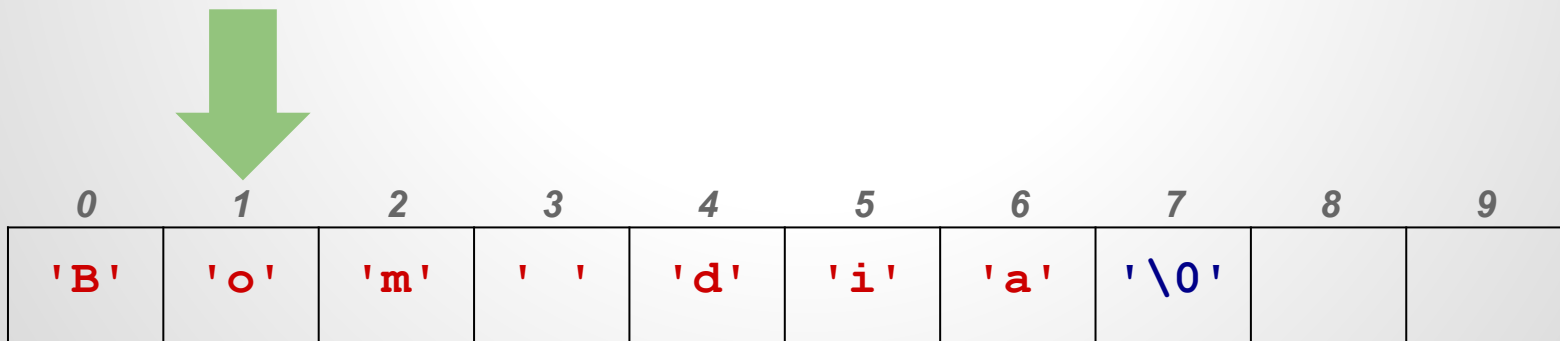
# Tamanho de um texto

- No começo o tamanho é: zero

**tam = 2**

Percorra o seu texto começando da posição inicial enquanto não encontrar o '`\0`'.

Cada novo caractere identifica uma letra (some 1 a cada posição percorrida).




# Tamanho de um texto

- No começo o tamanho é: zero

**tam = 3**

Percorra o seu texto começando da posição inicial enquanto não encontrar o '**\0**'.

Cada novo caractere identifica uma letra (some 1 a cada posição percorrida).



0	1	2	3	4	5	6	7	8	9
'B'	'o'	'm'	' '	'd'	'i'	'a'	'\0'		

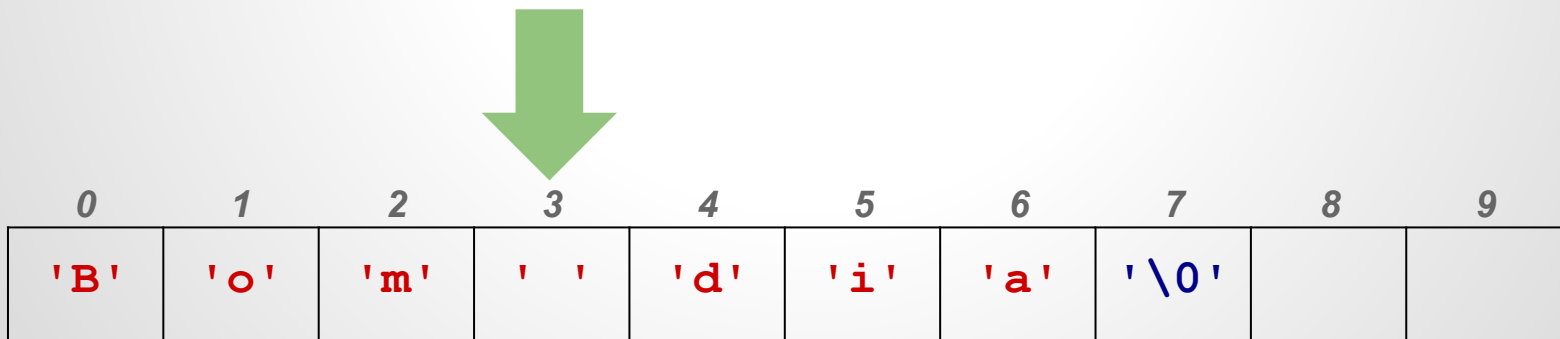
# Tamanho de um texto

- No começo o tamanho é: zero

**tam = 4**

Percorra o seu texto começando da posição inicial enquanto não encontrar o '**\0**'.

Cada novo caractere identifica uma letra (some 1 a cada posição percorrida).






# Tamanho de um texto

- No começo o tamanho é: zero

**tam = 5**

Percorra o seu texto começando da posição inicial enquanto não encontrar o '`\0`'.

Cada novo caractere identifica uma letra (some 1 a cada posição percorrida).



0	1	2	3	4	5	6	7	8	9
'B'	'o'	'm'	' '	'd'	'i'	'a'	'\0'		

# Tamanho de um texto

- No começo o tamanho é: zero

**tam = 6**

Percorra o seu texto começando da posição inicial enquanto não encontrar o '`\0`'.

Cada novo caractere identifica uma letra (some 1 a cada posição percorrida).



0	1	2	3	4	5	6	7	8	9
'B'	'o'	'm'	' '	'd'	'i'	'a'	'\0'		

# Tamanho de um texto

- No começo o tamanho é: zero

**tam = 7**

Percorra o seu texto começando da posição inicial enquanto não encontrar o '**\0**'.

Cada novo caractere identifica uma letra (some 1 a cada posição percorrida).



0	1	2	3	4	5	6	7	8	9
'B'	'o'	'm'	' '	'd'	'i'	'a'	'\0'		

# Tamanho de um texto

- No começo o tamanho é: zero

**tam = 7**

Percorra o seu texto começando da posição inicial enquanto não encontrar o '**\0**'.

Cada novo caractere identifica uma letra (some 1 a cada posição percorrida).

Hora de parar!

0	1	2	3	4	5	6	7	8	9
'B'	'o'	'm'	' '	'd'	'i'	'a'	'\0'		

# Tamanho de um texto

```
#include <stdio.h>
int main() {
    char texto[200];
    int tam;

    printf("Informe algum texto: ");
    scanf("%[^\n]", texto);

    /* FOR sem bloco. Veja o ";" */
    for (tam = 0; texto[tam] != '\0'; tam += 1);

    printf("O tamanho do texto eh %d\n", tam);

    return 0;
}
```

# Tamanho de um texto <string.h>

```
#include <stdio.h>
#include <string.h>
int main() {
    char texto[200];
    int tam;

    printf("Informe algum texto: ");
    scanf("%[^\\n]", texto);

    tam = strlen(texto);

    printf("O tamanho do texto eh %d\\n", tam);

    return 0;
}
```

# Tamanho de um texto <string.h>

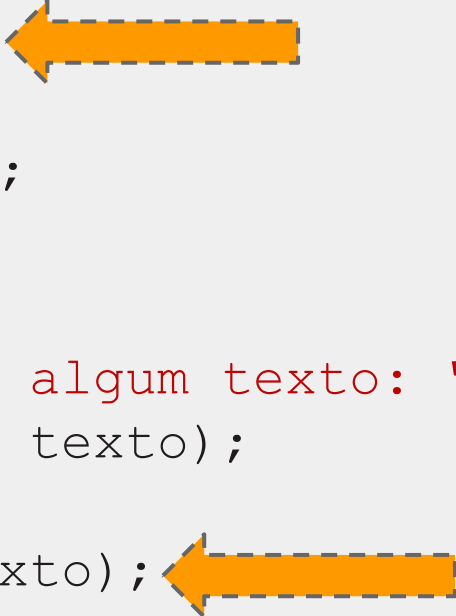
```
#include <stdio.h>
#include <string.h>
int main() {
    char texto[200];
    int tam;

    printf("Informe algum texto: ");
    scanf("%[^\\n]", texto);

    tam = strlen(texto);

    printf("O tamanho do texto eh %d\\n", tam);

    return 0;
}
```



# Exemplo

- Crie um programa para contar a quantidade de **espaços** em um texto



# Contar quantidade de espaços

- No começo a quantidade é: zero

Percorra o seu texto começando da posição inicial enquanto não encontrar o `'\0'`.

Cada novo `' '` identifica um espaço (conte mais 1 quanto encontrar).

0	1	2	3	4	5	6	7	8	9
'A'	' '	'T'	'a'	' '	'B'	'o'	'm'	'!'	'\0'

# Contar quantidade de espaços

- No começo a quantidade é: zero

**i = 0**

**qtde = 0**

Percorra o seu texto começando da posição inicial enquanto não encontrar o '**\0**'.

Cada novo ' ' identifica um espaço (conte mais 1 quando encontrar).

0	1	2	3	4	5	6	7	8	9
'A'	' '	'T'	'a'	' '	'B'	'o'	'm'	'!'	'\0'

# Contar quantidade de espaços

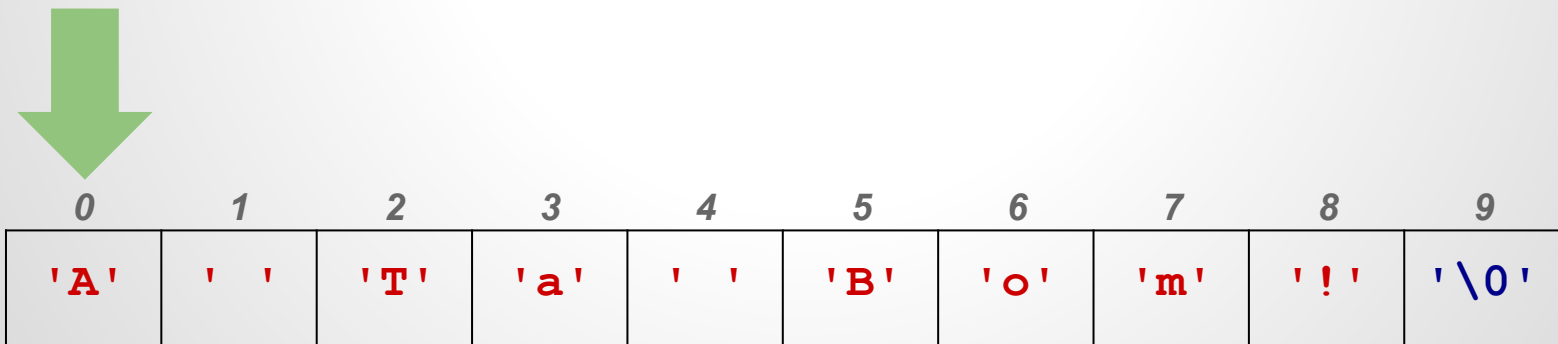
- No começo a quantidade é: zero

**i = 0**

**qtde = 0**

Percorra o seu texto começando da posição inicial enquanto não encontrar o '**\0**'.

Cada novo ' ' identifica um espaço (conte mais 1 quando encontrar).



# Contar quantidade de espaços

- No começo a quantidade é: zero

**i = 1**

**qtde = 1**

Percorra o seu texto começando da posição inicial enquanto não encontrar o '**\0**'.

Cada novo ' ' identifica um espaço (conte mais 1 quando encontrar).



0	1	2	3	4	5	6	7	8	9
'A'	' '	'T'	'a'	' '	'B'	'o'	'm'	'!'	'\0'

# Contar quantidade de espaços

- No começo a quantidade é: zero

**i = 2**

**qtde = 1**

Percorra o seu texto começando da posição inicial enquanto não encontrar o '**\0**'.

Cada novo ' ' identifica um espaço (conte mais 1 quando encontrar).



0	1	2	3	4	5	6	7	8	9
'A'	' '	'T'	'a'	' '	'B'	'o'	'm'	'!'	'\0'

# Contar quantidade de espaços


- No começo a quantidade é: zero

**i = 3**

**qtde = 1**

Percorra o seu texto começando da posição inicial enquanto não encontrar o '**\0**'.

Cada novo ' ' identifica um espaço (conte mais 1 quando encontrar).



0	1	2	3	4	5	6	7	8	9
'A'	' '	'T'	'a'	' '	'B'	'o'	'm'	'!'	'\0'


# Contar quantidade de espaços

- No começo a quantidade é: zero

**i = 4**

**qtde = 2**

Percorra o seu texto começando da posição inicial enquanto não encontrar o '**\0**'.  
Cada novo ' ' identifica um espaço (conte mais 1 quando encontrar).



0	1	2	3	4	5	6	7	8	9
'A'	' '	'T'	'a'	' '	'B'	'o'	'm'	'!'	'\0'

# Contar quantidade de espaços

- No começo a quantidade é: zero

**i = 5**

**qtde = 2**

Percorra o seu texto começando da posição inicial enquanto não encontrar o '**\0**'.  
Cada novo ' ' identifica um espaço (conte mais 1 quando encontrar).



0	1	2	3	4	5	6	7	8	9
'A'	' '	'T'	'a'	' '	'B'	'o'	'm'	'!'	'\0'



# Contar quantidade de espaços

- No começo a quantidade é: zero

**i = 6**

**qtde = 2**

Percorra o seu texto começando da posição inicial enquanto não encontrar o '**\0**'.  
Cada novo ' ' identifica um espaço (conte mais 1 quando encontrar).



0	1	2	3	4	5	6	7	8	9
'A'	' '	'T'	'a'	' '	'B'	'o'	'm'	'!'	'\0'

# Contar quantidade de espaços

- No começo a quantidade é: zero

**i = 7**

**qtde = 2**

Percorra o seu texto começando da posição inicial enquanto não encontrar o '**\0**'.  
Cada novo ' ' identifica um espaço (conte mais 1 quando encontrar).



0	1	2	3	4	5	6	7	8	9
'A'	' '	'T'	'a'	' '	'B'	'o'	'm'	'!'	'\0'

# Contar quantidade de espaços

- No começo a quantidade é: zero

**i = 8**

**qtde = 2**

Percorra o seu texto começando da posição inicial enquanto não encontrar o '**\0**'.  
Cada novo ' ' identifica um espaço (conte mais 1 quando encontrar).



0	1	2	3	4	5	6	7	8	9
'A'	' '	'T'	'a'	' '	'B'	'o'	'm'	'!'	'\0'

# Contar quantidade de espaços

- No começo a quantidade é: zero

**i = 9**

**qtde = 2**

Percorra o seu texto começando da posição inicial enquanto não encontrar o '**\0**'.

Cada novo ' ' identifica um espaço (conte mais 1 quando encontrar).

Hora de parar!



0	1	2	3	4	5	6	7	8	9
'A'	' '	'T'	'a'	' '	'B'	'o'	'm'	'!'	'\0'

# Contar quantidade de espaços

- No começo a quantidade é: zero

**i = 9**

**qtde = 2**



Percorra o seu texto começando da posição inicial enquanto não encontrar o '`\0`'.

Cada novo ' ' identifica um espaço (conte mais 1 quando encontrar).

Hora de parar!



0	1	2	3	4	5	6	7	8	9
'A'	' '	'T'	'a'	' '	'B'	'o'	'm'	'!'	'\0'

# Contar quantidade de espaços

```
#include <stdio.h>
int main() {
    char texto[200];
    int i, qtde;

    printf("Informe algum texto: ");
    scanf("%[^\n]", texto);

    for (i = 0, qtde = 0; texto[i] != '\0'; i += 1) {
        if (texto[i] == ' '){
            qtde += 1;
        }
    }

    printf("O texto possui %d espacos.\n", qtde);
    return 0;
}
```

# Exemplo

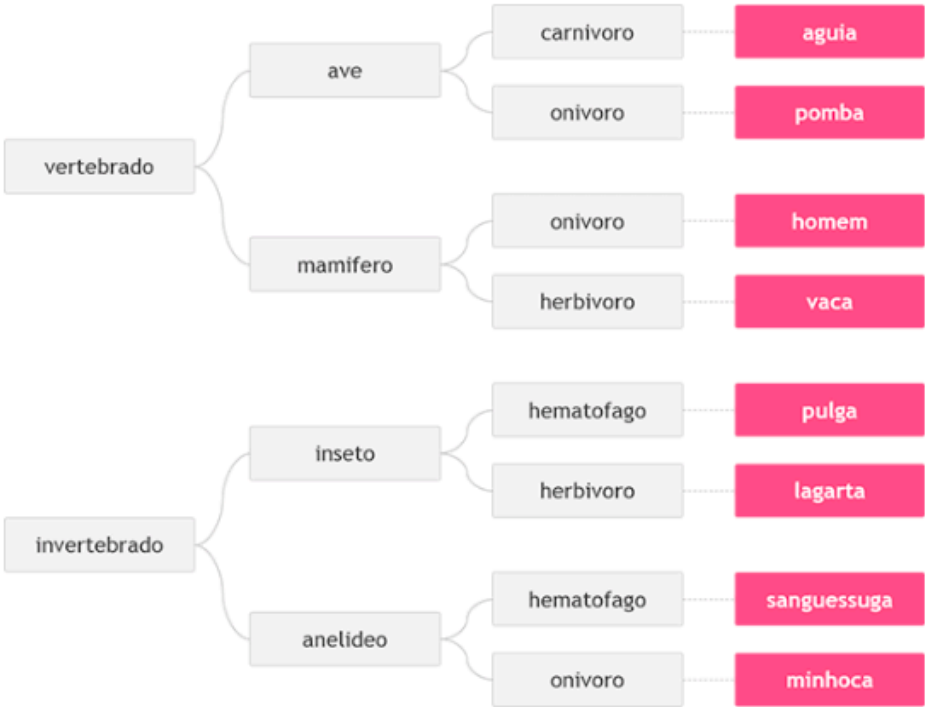
- Crie um programa para contar a quantidade de **espaços** em um texto
- Adapte este programa para contar a quantidade de **qualquer** caractere em um texto

**Fazer este!!!**

**Dúvidas?**



Neste problema, você deverá ler 3 palavras que definem o tipo de animal possível segundo o esquema abaixo, da esquerda para a direita. Em seguida conclua qual dos animais seguintes foi escolhido, através das três palavras fornecidas.



**Entrada**

A entrada contém 3 palavras, uma em cada linha, necessárias para identificar o animal segundo a figura acima, com todas as letras minúsculas.

**Saída**

Imprima o nome do animal correspondente à entrada fornecida.

Exemplos de Entrada	Exemplos de Saída
vertebrado mamifero onivoro	homem