

BCC 201 - Introdução à Programação I

# Cadeia de caracteres (*strings*)

Guillermo Cámara-Chávez  
UFOP

# Cadeia de caracteres I

- ▶ Definição: são seqüências de letras, números ou símbolos onde o último caracter é o caracter nulo ( $\backslash 0$ )
- ▶ Na linguagem C utilizamos vetores de tipo `char` para armazenar cadeias de caracteres.

# Cadeia de caracteres II

- ▶ Por exemplo, para declarar um espaço de memória que contenha 20 caracteres fazemos:

```
char nome[20];
```



```
scanf("%s", nome);
```



- ▶ A função `scanf`, quando utilizada com o parâmetro `%s` lê todos os caracteres até encontrar um caracter igual a espaço ou fim de linha

# Cadeia de caracteres III

- ▶ Este é o estilo de *string* usado pela linguagem C pura.
- ▶ Para manipular este tipo de *string* é preciso ter certo cuidado (se **acessamos um endereço fora do vetor**, invadiremos outras áreas da memória).

## Cadeia de caracteres IV

- ▶ Pergunta: Qual o tamanho do menor vetor que devemos declarar para armazenar uma cadeia de caracteres com 10 letras?

## Cadeia de caracteres V

- ▶ Escreva um program que lê uma palavra da entrada e imprime o número de caracteres desta palavra

# Cadeia de caracteres VI

```
#include <stdio.h>
int main()
{
    char vetor[100];
    int i, n;
    printf("Entre com a palavra: ");
    scanf("%s", vetor);
    i = 0;
    n = 0;
    while(vetor[i++] != '\0')
        n++;
    printf("O numero de caracteres eh %d", n);
    return 0;
}
```

# Cadeia de caracteres VII

## ► Funções Importantes (string.h)

- `strlen`: retorna o número de caracteres da cadeia de caracteres, sem contar o `'\0'`. Ex,

```
strlen("casa")
```

- `strcmp`: compara caracter a caracter duas cadeias e retorna o resultado dessa comparação. Retorna um valor inteiro
  - `= 0`: ambos strings são iguais
  - `> 0`: o primeiro caracter que não casa tem um valor superior na primeira cadeia
  - `< 0`: o primeiro caracter que não casa tem um valor inferior na primeira cadeia



## Cadeia de caracteres VIII

- ▶ `strcmp("casa", "carro")` retorna 1
- ▶ `strcmp("casa", "casa")` retorna 0
- ▶ `strcmp("carro", "casa")` retorna -1

## Cadeia de caracteres IX

- ▶ `strcmpi`: Compara caracter a caracter duas cadeias ignorando se as letras são maiúsculas ou minúsculas, e retorna o resultado dessa comparação. Ex.:
  - ▶ `strcmpi("Casa", "carro")`, retorna 1
  - ▶ `strcmpi("Casa", "cASa")`, retorna 0
  - ▶ `strcmpi("caRro", "casA")`, retorna -1
- ▶ `strcpy`: copia uma string para dentro de outra. Ex.:

```
strcpy(vetor, "carro");  
strcpy(vetor2, vetor);
```

# Cadeia de caracteres X

- ▶ `strcat`: concatena duas string, a primeira string recebe o seu conteúdo seguido da segunda string. Ex.:

```
strcpy(vetor , "uva + ");  
strcat(vetor , "banana +");  
strcat(vetor , "pera");
```

# Cadeia de caracteres XI

- ▶ Criar a função que receba um string e que gere outro *string* com a ordem inversa dos caracteres. Ex “alem” deve ser gerado “mela”.

## Cadeia de caracteres XII

```
#include <string.h>
void Inverte(char*, char*, int);
int main()
{
    char cad1[100] = "", cad2[100] = "";
    int i, j;
    printf("Inserir uma cadeia");
    scanf("%s", cad1);
    Inverte(cad1, cad2, 100);
    printf("cadeia1: %s cadeia2: %s", cad1, cad2);
    return 0;
}
```

## Cadeia de caracteres XIII

```
void Inverte(char* cad, char* cadinv, int tamCadInv)
{
    int n = strlen(cad), i, j;
    if (n <= tamCadInv){
        for (i = 0, j = n-1; i < n ; i++; j--)
            cadinv[i] = cad[j];
        cadinv[i] = '\0';
    }
}
```

## Cadeia de caracteres XIV

- ▶ Implementar uma função que crie uma cópia de um string

## Cadeia de caracteres XV

```
void Copia(char cadCopia[], char cad[], int n);
int main()
{
    char cad[50], cad_copia[50];
    printf("Inserir uma cadeia \n");
    scanf("%s", cad);
    Copia(cadCopia, cad, 50);
    printf("cadeia: %s copia: %s", cad, cadCopia);
    return 0;
}
```



## Cadeia de caracteres XVI

```
void Copia(char cadCopia[], char cad[], int n)
{
    int tamCad = strlen(cad), i;
    if (tamCad <= n){
        for (i = 0; i <= tamCad; i++)
        {
            cadCopia[i] = cad[i];
        }
    }
}
```

## Cadeia de caracteres XVII

- ▶ Criar uma função que elimine os espaços em branco de uma cadeia de caracteres

# Cadeia de caracteres XVIII

```
void eliminaBraco(char*, char* , int n);  
int main()  
{  
    char cad[50], cadSem[50];  
    printf("Inserir uma cadeia \n");  
    fgets(cad, 50, stdin);  
    eliminaBraco(cadSem, cad, 50);  
    printf("cadeia: %s ", cadSem);  
    return 0;  
}
```

# Cadeia de caracteres XIX

```
void eliminaBraco(char* cadSem, char* cad, int n)
{
    int tamCad = strlen(cad), i, j;
    j = 0;
    if (tamCad <= n)
    {
        for (i = 0; i < tamCad; i++)
            if (cad[i] != ' ')
                cadSem[j++] = cad[i];
    }
    cadSem[j] = '\0';
}
```

## Cadeia de caracteres XX

- ▶ Criar as funções, LEFT e RIGHT, que retornem a subcadeia de  $n$  elementos a partir da esquerda e da direita respectivamente. Ex cad = "transformar". Mostrar os 3 primeiros caracteres a partir da esquerda: "tra". Mostrar os 5 primeiros caracteres a partir da direita: "ormar".

# Cadeia de caracteres XXI

```
void Left(char cadLeft[], char cad[], int n);
void Right(char cadRight[], char cad[], int n);
int main()
{
    char cad1[50], cadRight[50], cadLeft[50];
    printf("Inserir cadeia: ");
    scanf("%s", cad1);
    Left(cadLeft, cad1, 4);
    Right(cadRight, cad1, 4);
    printf("Right %s \n", cadRight);
    printf("Left %s \n", cadLeft);
    return 0;
}
```

## Cadeia de caracteres XXII

```
void Left(char cadLeft[], char cad[], int n)
{
    int tamCad = strlen(cad), i;
    if (n <= tamCad)
    {
        for (i = 0; i < n; i++)
            cadLeft[i] = cad[i];
        cadLeft[i] = '\0';
    }
}
```

## Cadeia de caracteres XXIII

```
void Right(char cadRight[], char cad[], int n)
{
    int tamCad = strlen(cad), i, j;
    if (n <= tamCad)
    {
        j = tamCad-n;
        for (i = 0; i < n; i++,j++)
            cadRight[i] = cad[j];
        cadRight[i] = '\0';
    }
}
```



## Cadeia de caracteres XXIV

- ▶ O método `substr(ini_pos, numCaract)` da classe `string` em C++ extrai uma subcadeia de dimensão *numCaract* a partir da posição `ini_pos`. Por exemplo,

```
string palavra = "a maioria voltara";  
cad = palavra.substr(2, 7);
```

A variável `cad` contém a cadeia “maioria”.

Dita função não está implementada na linguagem C.

Implementar a função que execute essa mesma operação.

# Cadeia de caracteres XXV

```
void substr(char subcad[], char cad[], int ini, int n);  
int main()  
{  
    char cad[50], cadSem[50];  
    printf("Inserir uma cadeia \n");  
    gets(cad);  
    substr(cadSem, cad, 2, 4);  
    printf("cadeia: %s ", cadSem);  
    return 0;  
}
```

## Cadeia de caracteres XXVI

```
void substr(char subcad[], char cad[], int ini, int n)
{
    int tamCad = strlen(cad), i, j = 0;
    if (ini+n-1 < tamCad)
    {
        for (i = ini; i <= ini+n-1; i++)
            subcad[j++] = cad[i];
    }
    subcad[j] = '\n';
}
```

## Cadeia de caracteres XXVII

Determinação da primeira ocorrência de uma subcadeia em uma cadeia.

# Cadeia de caracteres XXVIII

frase

a	l	o		a	l	u	n	o		n	o	\0			
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15

pal

a	l	u	n	o	\0				
0	1	2	3	4	5	6	7	8	9



# Cadeia de caracteres XXIX

frase

a	l	o		a	l	u	n	o		n	o	\0			
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15

pal

a	l	u	n	o	\0				
0	1	2	3	4	5	6	7	8	9



# Cadeia de caracteres XXX

a	l	o		a
---	---	---	--	---

0 1 2 3 4

frase

a	l	o		a	l	u	n	o		n	o	\0			
---	---	---	--	---	---	---	---	---	--	---	---	----	--	--	--

0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15

pal

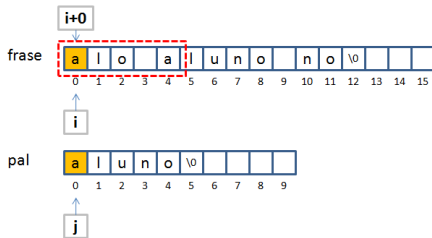
a	l	u	n	o	\0				
---	---	---	---	---	----	--	--	--	--

0 1 2 3 4 5 6 7 8 9

# Cadeia de caracteres XXXI



a	l	o		a
0	1	2	3	4

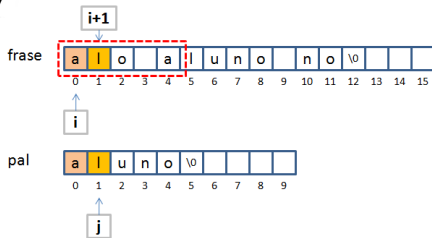




# Cadeia de caracteres XXXII



a	l	o		a
0	1	2	3	4

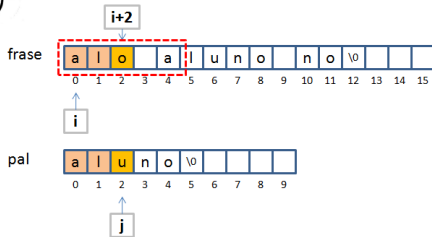


# Cadeia de caracteres XXXIII



a	l	o		a
---	---	---	--	---

0 1 2 3 4



# Cadeia de caracteres XXXIV

l	o		a	l
---	---	--	---	---

0 1 2 3 4

frase

a	l	o		a	l	u	n	o		n	o	\0			
---	---	---	--	---	---	---	---	---	--	---	---	----	--	--	--

0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15

↑  
i

pal

a	l	u	n	o	\0				
---	---	---	---	---	----	--	--	--	--

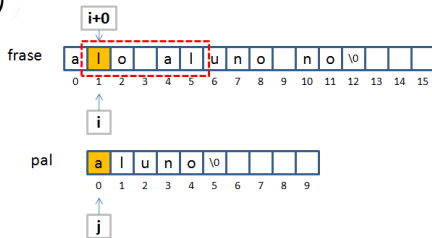
0 1 2 3 4 5 6 7 8 9

↑  
j

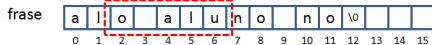
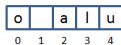
# Cadeia de caracteres XXXV



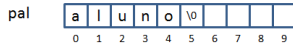
l	o		a	l
0	1	2	3	4



# Cadeia de caracteres XXXVI

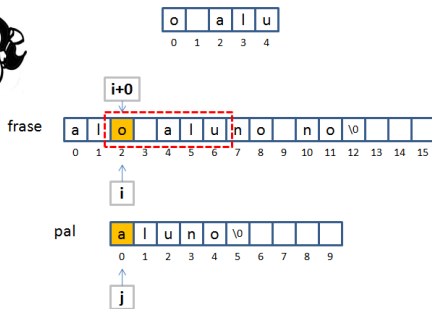


↑  
i



↑  
j

# Cadeia de caracteres XXXVII



# Cadeia de caracteres XXXVIII

	a	l	u	n
--	---	---	---	---

0 1 2 3 4

frase

a	l	o		a	l	u	n	o		n	o	\0			
---	---	---	--	---	---	---	---	---	--	---	---	----	--	--	--

0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15

↑  
**i**

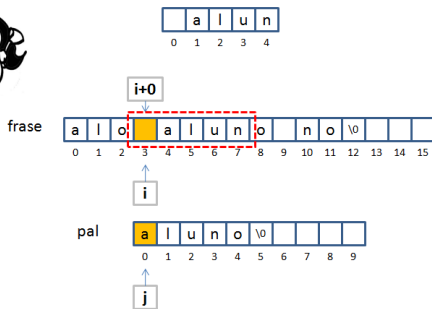
pal

a	l	u	n	o	\0				
---	---	---	---	---	----	--	--	--	--

0 1 2 3 4 5 6 7 8 9

↑  
**j**

# Cadeia de caracteres XXXIX





# Cadeia de caracteres XL

a	l	u	n	o
---	---	---	---	---

0 1 2 3 4

frase

a	l	o	a	l	u	n	o	n	o	\0				
---	---	---	---	---	---	---	---	---	---	----	--	--	--	--

0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15

i

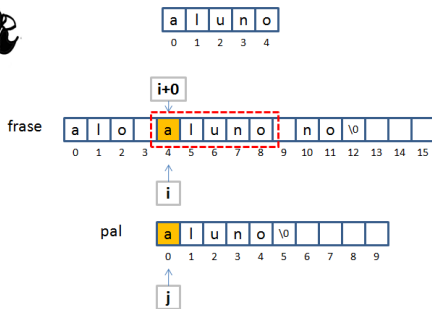
pal

a	l	u	n	o	\0				
---	---	---	---	---	----	--	--	--	--

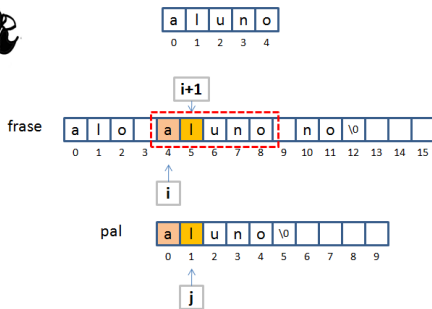
0 1 2 3 4 5 6 7 8 9

j

# Cadeia de caracteres XLI



# Cadeia de caracteres XLII



# Cadeia de caracteres XLIII



a	l	u	n	o
---	---	---	---	---

0 1 2 3 4

i+2

frase

a	l	o	a	l	u	n	o	n	o	\0				
---	---	---	---	---	---	---	---	---	---	----	--	--	--	--

0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15

i

pal

a	l	u	n	o	\0				
---	---	---	---	---	----	--	--	--	--

0 1 2 3 4 5 6 7 8 9

j

# Cadeia de caracteres XLIV



a	l	u	n	o
0	1	2	3	4

i+3

frase

a	l	o	a	l	u	n	o	n	o	\0					
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15

i

pal

a	l	u	n	o	\0				
0	1	2	3	4	5	6	7	8	9

j

# Cadeia de caracteres XLV



a	l	u	n	o
0	1	2	3	4

i+4

frase

a	l	o	a	l	u	n	o	n	o	\0					
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15

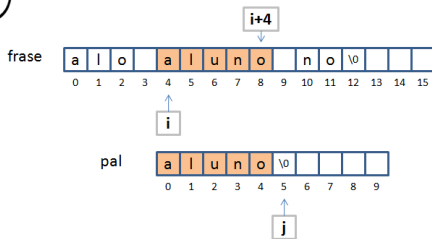
i

pal

a	l	u	n	o	\0				
0	1	2	3	4	5	6	7	8	9

j

# Cadeia de caracteres XLVI



# Cadeia de caracteres XLVII

l	u	n	o	
---	---	---	---	--

0 1 2 3 4

frase

a	l	o	a	l	u	n	o		n	o	\0				
---	---	---	---	---	---	---	---	--	---	---	----	--	--	--	--

0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15

↑  
i

pal

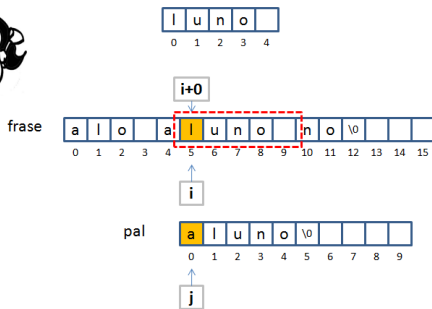
a	l	u	n	o	\0				
---	---	---	---	---	----	--	--	--	--

0 1 2 3 4 5 6 7 8 9

↑  
j



# Cadeia de caracteres XLVIII



# Cadeia de caracteres XLIX

u	n	o		n
---	---	---	--	---

0 1 2 3 4

frase

a	l	o		a	l	u	n	o		n	o	\0			
---	---	---	--	---	---	---	---	---	--	---	---	----	--	--	--

0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15

↑  
i

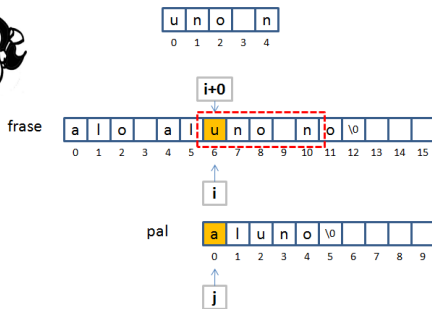
pal

a	l	u	n	o	\0				
---	---	---	---	---	----	--	--	--	--

0 1 2 3 4 5 6 7 8 9

↑  
j

# Cadeia de caracteres L



# Cadeia de caracteres LI

n	o		n	o
---	---	--	---	---

0 1 2 3 4

frase

a	l	o		a	l	u	n	o		n	o	\0			
---	---	---	--	---	---	---	---	---	--	---	---	----	--	--	--

0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15

i

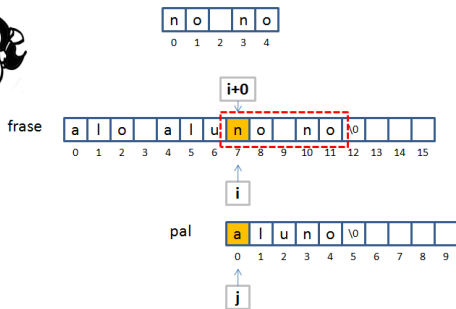
pal

a	l	u	n	o	\0				
---	---	---	---	---	----	--	--	--	--

0 1 2 3 4 5 6 7 8 9

j

# Cadeia de caracteres LII



# Cadeia de caracteres LIII

frase

a	l	o		a	l	u	n	o		n	o	\0			
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15

i

pal

a	l	u	n	o	\0				
0	1	2	3	4	5	6	7	8	9

j



# Cadeia de caracteres LIV

```
int EstaContida(char cad[], char subcad[]);  
int main()  
{  
    char cad1[50], cad2[50];  
    printf("Inserir cadeia e subcadeia: ");  
    scanf("%s %s", cad1, cad2);  
    if (EstaContida(cad1, cad2))  
        printf("subcadeia contida");  
    return 0;  
}
```

# Cadeia de caracteres LV

```
int EstaContida(char cad[], char subcad[]){
    int tamCad1, tamCad2, i, j, cont;
    tamCad1 = strlen(cad);
    tamCad2 = strlen(subcad);
    if (tamCad2 <= tamCad1){
        for (i = 0; i < tamCad1-tamCad2; i++)
        {
            for (j = 0, cont = 0; j < tamCad2;
                j++, cont++)
            {
                if (cad[i+j] != subcad[j])
                    break;
            }
            if (cont == tamCad2)
                return 1;
        }
    }
    return 0;
}
```



FIM