Cadeias de Caracteres

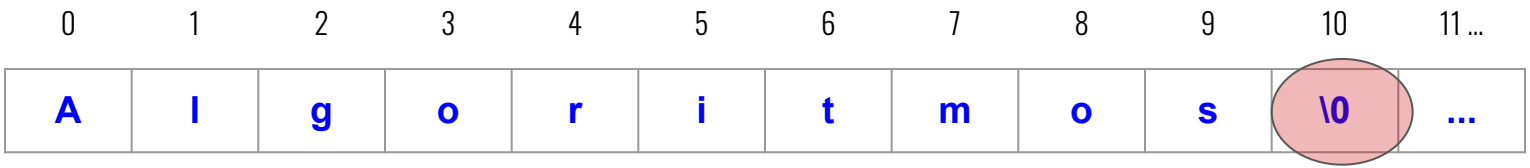
Funções e Manipulação de Strings

Algoritmos e Programação
Carlos Michel Betemps

Cadeias de Caracteres (strings) - declaração e inicialização

String é o nome dado às variáveis que mantém uma sequência de caracteres adjacentes na memória do computador.

```
//inicializando uma string (cadeia de caracteres)
char str[50] = "Algoritmos";
//ou
char str[50] = {'A','l','g','o','r','i','t','m','o','s'};
```



String

```
clang-7 -pthread -lm -o main main.c
./main
Digite uma string: Algoritmos
strcpy: [Algoritmos] [Algoritmos]
strcmp_1: [0]
strcmp_2: [-15]
strcmp_3: [15]
strlen: [Algoritmos] 10 caracteres
AlgoritmosAlgoritmos
```

```c
#include <stdio.h>
#include <string.h>
int main(void) {
 char str2[50]="Programação",str1[50], str[50];
 printf("Digite uma string: ");
 scanf(" %[^\n]", str1);
 strcpy(str,str1);
 printf("strcpy: [%s] [%s]\n", str, str1);
 if (strcmp(str,str1) == 0) printf("strcmp 1: [%d]\n", strcmp(str,str1));
 printf("strcmp 2: [%d]\n", strcmp(str,str2));
 printf("strcmp 3: [%d]\n", strcmp(str2,str));
 printf("strlen: [%s] %ld caracteres\n", str, strlen(str));
 strcat(str1,str);
 printf("%s\n", str1);
 return 0;
}
```

String

```c
#include <stdio.h>
#include <string.h>
int main(void) {
 char str2[50]="Programação",str1[50], str[50];
 printf("Digite uma string: ");
 scanf(" %[^\n]", str1);
 strcpy(str,str1);
 printf("strcpy: [%s] [%s]\n", str, str1);
 if (strcmp(str,str1) == 0) printf("strcmp 1: [%d]\n", strcmp(str,str1));
 printf("strcmp 2: [%d]\n", strcmp(str,str2));
 printf("strcmp 3: [%d]\n", strcmp(str2,str));
 printf("strlen: [%s] %ld caracteres\n", str, strlen(str));
 strcat(str1,str);
 printf("%s\n", str1);
 return 0;
}
```

leitura de uma string usando *scanf*:

- `" %[^\n]"`: permite a leitura de uma string contendo espaços (*ex. um nome completo, uma frase*).
- o comando ***gets(str)*** também poderia ser utilizado para receber uma string via teclado

String

```c
#include <stdio.h>
#include <string.h>
int main(void) {
 char str2[50]="Programação",str1[50], str[50];
 printf("Digite uma string: ");
 scanf(" %[^\n]", str1);
 strcpy(str,str1);
 printf("strcpy: [%s] [%s]\n", str, str1);
 if (strcmp(str,str1) == 0) printf("strcmp 1: [%d]\n", strcmp(str,str1));
 printf("strcmp 2: [%d]\n", strcmp(str,str2));
 printf("strcmp 3: [%d]\n", strcmp(str2,str));
 printf("strlen: [%s] %ld caracteres\n", str, s
 strcat(str1,str);
 printf("%s\n", str1);
 return 0;
}
```

strcpy(destino, origem): permite a cópia da string mantida no parâmetro *origem* para o parâmetro *destino*.

```
> clang-7 -pthread -lm -o main main.c
> ./main
Digite uma string: Algoritmos
strcpy: [Algoritmos] [Algoritmos]
strcmp_1: [0]
strcmp_2: [-15]
strcmp_3: [15]
strlen: [Algoritmos] 10 caracteres
AlgoritmosAlgoritmos
>
```

String

```
> clang-7 -pthread -lm -o main main.c
> ./main
Digite uma string: Algoritmos
strcpy: [Algoritmos] [Algoritmos]
strcmp_1: [0]
strcmp_2: [-15]
strcmp_3: [15]
strlen: [Algoritmos] 10 caracteres
AlgoritmosAlgoritmos
>
```

```c
#include <stdio.h>
#include <string.h
int main(void) {
 char str2[50]="Pr
 printf("Digite uma string: ");
 scanf(" %[^\n]", str1);
 strcpy(str,str1);
 printf("strcpy: [%s] [%s]\n", str, str1);
 if (strcmp(str,str1) == 0) printf("strcmp 1: [%d]\n", strcmp(str,str1));
 printf("strcmp 2: [%d]\n", strcmp(str,str2));
 printf("strcmp 3: [%d]\n", strcmp(str2,str));
 printf("strlen: [%s] %ld caracteres\n", str, strlen(str));
 strcat(str1,str);
 printf("%s\n", str1);
 return 0;
}
```

strcmp(s1, s2): permite comparar lexicograficamente (alfabeticamente) comparar a string *s1* e a string *s2*.

String

```
clang-7 -pthread -lm -o main main.c
./main
Digite uma string: Algoritmos
strcpy: [Algoritmos] [Algoritmos]
strcmp_1: [0]
strcmp_2: [-15]
strcmp_3: [15]
strlen: [Algoritmos] 10 caracteres
AlgoritmosAlgoritmos
```

```c
#include <stdio.h>
#include <string.h>
int main(void)
 char str2[50]="Algoritmos";
 printf("Digite uma string: ");
 scanf(" %[^\n]", str1);
 strcpy(str,str1);
 printf("strcpy: [%s] [%s]\n", str, str1);
 if (strcmp(str,str1) == 0) printf("strcmp 1: [%d]\n", strcmp(str,str1));
 printf("strcmp 2: [%d]\n", strcmp(str,str2));
 printf("strcmp 3: [%d]\n", strcmp(str2,str));
 printf("strlen: [%s] %ld caracteres\n", str, strlen(str));
 strcat(str1,str);
 printf("%s\n", str1);
 return 0;
}
```

int: strlen(s): retorna o tamanho de uma string (quantos caracteres tem a string)

String

```c
#include <stdio.h>
#include <string.h>
int main(void) {
 char str2[50]="Programação",str1[50], str[50];
 printf("Digite uma string: ");
 scanf(" %[^\n]", str1);
 strcpy(str,str1);
 printf("strcpy: [%s] [%s]\n", str, str1);
 if (strcmp(str,str1) == 0) printf("strcmp 1: [%d]\n", strcmp(str,str1));
 printf("strcmp 2: [%d]\n", strcmp(str,str2));
 printf("strcmp 3: [%d]\n", strcmp(str2,str));
 printf("strlen: [%s] %ld caracteres\n", str, strlen(str));
 strcat(str1,str);
 printf("%s\n", str1);
 return 0;
}
```

strcat(s1, s2): concatena a string *s2* ao final da string *s1*.

String

```
> clang-7 -pthread -lm -o main main.c
> ./main
Digite uma string: Algoritmos
strcpy: [Algoritmos] [Algoritmos]
strcmp_1: [0]
strcmp_2: [-15]
strcmp_3: [15]
strlen: [Algoritmos] 10 caracteres
AlgoritmosAlgoritmos
>
```

```c
#include <stdio
#include <strin
int main(void)
 char str2[50]="
 printf("Digite uma string: ");
 scanf(" %[^\n]", str1);
 strcpy(str,str1);
 printf("strcpy: [%s] [%s]\n", str, str1);
 if (strcmp(str,str1) == 0) printf("strcmp 1: [%d]\n", strcmp(str,str1));
 printf("strcmp 2: [%d]\n", strcmp(str,str2));
 printf("strcmp 3: [%d]\n", strcmp(str2,str));
 printf("strlen: [%s] %ld caracteres\n", str, strlen(str));
 strcat(str1,str);
 printf("%s\n", str1);
 return 0;
}
```

- Concatena imprimindo a string *str1* já concatenada.
- Este comando poderia ser substituído pelo comando de impressão de strings (puts)
  - puts(str1);

String - acessando um caractere da string

```c
#include <stdio.h>
#include <string.h>
int main(void) {
 char str[50]="Algoritmos";
 for(int i=0; i< strlen(str);i++){
   printf("%c\n", str[i]);
 }
 return 0;
}
```

```
> clang-7 -pthread -lm -o main main.c
> ./main
A
l
g
o
r
i
t
m
o
s
>
```

Entre colchetes indica-se a posição do caractere que se quer acessar - posições iniciam no valor zero (0)

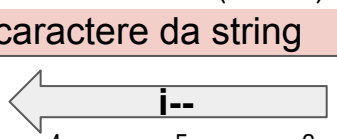
```c
/*Imprimindo cada caractere de uma string linha a linha.*/
```

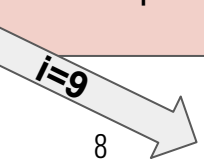
String - acessando um caractere da string

```
#include <stdio.h>
#include <string.h>
int main(void) {
 char str[50]="Algoritmos";
 for(int i=strlen(str)-1; i>=0; i--){
   printf("%c", str[i]);
 }
 printf("\n");
 return 0;
}
/*Imprimindo uma string ao contrário.*/
```

```
> clang-7 -pthread -lm -o main main.c
> ./main
somtiroglA
>
```

`i=strlen(str)-1:` i recebe a posição do último caractere da string





0	1	2	3	4	5	6	7	8	9	10	11 ...
A	l	g	o	r	i	t	m	o	s	\0	...

String - acessando um caractere da string

```
> clang-7 -pthread -lm -o main main.c
> ./main
Tamanho da String "Algoritmos": 10
>
```

```c
#include <stdio.h>
int main(void) {
 char str[50]="Algoritmos";
 int cont=0;
 while(str[cont] != '\0'){
   cont++;
 }
 printf("Tamanho da String \"%s\": %d\n", str, cont);
 return 0;
}
/*Calculando o número de caracteres (tamanho) de uma string sem
utilizar a strlen(s) da string.h.*/
```

`'\0'`: caractere nulo que marca o final de uma string. Enquanto não encontrar o caractere `'\0'` na string *str,* a variável *cont* vai sendo incrementada.

String - modificando caracteres da string

```
> clang-7 -pthread -lm -o main main
> ./main
Original: "Banana"
Modificada: "Batata"
>
```

```c
#include <stdio.h>
#include <string.h>
int main(void) {
 char str[50]="Banana";
 char C = 'n', N = 't';
 printf("Original: \"%s\"\n", str);
 for(int i=0; i<strlen(str); i++){
    if (str[i] == C){
      str[i] = N;
    }
 }
 printf("Modificada: \"%s\"\n", str);
 return 0;
}
/*Substituindo todas as ocorrências de um caractere C da String por
um novo caractere N*/
```

- Verifica se o caractere da string, na posição *i*, é igual a *C*.
  - Caso positivo, troca o caractere na posição *i* pelo caractere *N*.

Exercícios:

1. Faça um programa que leia uma string e a imprima de trás para a frente.
2. Faça um programa que leia uma string e a inverta. A string invertida deve ser armazenada na mesma variável. Em seguida, imprima a string invertida.
3. Faça um programa que leia uma string do teclado e que conte quantas vogais (a, e, i, o, u) ela possui. Entre com um caractere (vogal ou consoante) e substitua todas as vogais da palavra dada por esse caractere. Ao final, imprima a nova string e o número de vogais que ela possui.
4. Faça um programa que leia uma string e imprima uma mensagem dizendo se ela é um palíndromo ou não. Um palíndromo é uma palavra que tem a propriedade de poder ser lida tanto da direita para a esquerda como da esquerda para a direita. Exemplo: ovo, arara, rever, asa, osso, mussum.
5. Escreva um programa que leia uma string do teclado (contendo somente letras sem acentos) e que converta todos os seus caracteres em maiúscula. Dica: subtraia 32 dos caracteres cujo valor (código ASCII) está entre 97 e 122.
6. Escreva um programa que leia uma string do teclado (contendo somente letras sem acentos) e que converta todos os seus caracteres em minúscula. Dica: some 32 aos caracteres cujo valor (código ASCII) está entre 65 e 90.
7. Escreva um programa que recebe um string e dois valores inteiros não negativos i e j. Em seguida, imprima os caracteres contidos no segmento que vai de i a j da string s.
8. O código de César é uma das técnicas de criptografia mais simples e conhecidas. É um tipo de substituição no qual cada letra do texto é substituída por outra, que se apresenta n posições após ela no alfabeto. Por exemplo, com uma troca de três posições, a letra A seria substituída por D, B se tornaria E e assim por diante. Escreva um programa que faça uso desse código de César para três posições. Entre com uma string e imprima a string codificada. Exemplo:
    - String original: a ligeira raposa marrom saltou sobre o cachorro cansado
    - String codificada: d oljhlud udsrvd pduurp vdowrx vreuh r fdfkruur fdqvdgr