



# Vetor de caracteres (strings)

---

Prof. Tiago G. Botelho



# Strings em C++

---

- A linguagem C/C++ não possui suporte a strings! (mais ou menos...);
- Uma String em C/C++ nada mais é do que um vetor de caracteres;
- Mas o que é exatamente um caracter?



# Caracteres

---

- O caractere é um tipo de dados primitivo do C/C++, identificado pela palavra chave `char`;
- A forma literal de um caractere é o próprio caractere entre aspas simples:
  - **`char`** letraA = 'a';
- Para imprimir ou ler um caractere (família de funções `printf` e `scanf` se utiliza a sequência de controle `%c`;
  - `printf("%c\n", letraA);`

- `char s[10];`

[illegible]

- `s[0]='a';`

# S



# String

---

- Uma string é um array de caracteres;
- O que é exibido na tela?

```
1      #include<bits/stdc++.h>
2
3      main() {
4          char nome[20] = "jorge";
5          printf("%c\n", nome[0]);
6          printf("%c\n", nome[3]);
7          nome[3] = 'j';
8          printf("%s", nome);
9      }
10
```



# String

---

- Um string literal é representado por uma cadeia de caracteres delimitada por aspas duplas (como em "ABCD");
- Internamente, o string é terminado com o caracter '\0', que representa o número zero;



# Leitura e escrita de strings

---

- A biblioteca `<bits/stdc++.h>` disponibiliza as funções:
  - `gets()` //leitura
  - `puts()` //escrita





# gets(char \*palavra)

---

- Aguarda a digitação de uma sequência de caracteres no teclado pelo usuário, e copia todos eles para o vetor passado como argumento;
- A sequência deve terminar com ENTER;
- O ENTER **não** é copiado para o vetor;
- O caracter '\0' é inserido automaticamente depois do último caracter no vetor;
- Não é feita verificação de tamanho máximo na leitura.

# Exemplo de uso do gets

```
1  #include<bits/stdc++.h>
2
3  main() {
4      char palavra[20];
5      int i;
6      printf("Digite uma palavra");
7      gets(palavra);
8      printf("A palavra digitada e: %s",palavra);
9  }
```

■ char palavra[20];

	<i>0</i>	<i>1</i>	<i>2</i>	<i>3</i>	<i>4</i>	<i>5</i>	<i>6</i>	<i>...</i>	<i>18</i>	<i>19</i>
palavra	'L'	'P'	'1'	'\0'	?	?	?	...	?	?



```
scanf("^\\n",palavra);
```

---

- Também pode ser usado:

```
scanf("%xx^[\\n]s",palavra);
```

Aqui vai o  
número limite de  
caracteres lidos

- Alternativa para a função gets ();
- Especifica a quantidade máxima de caracteres que serão copiados da entrada para o vetor;
- Os demais caracteres digitados pelo usuário serão ignorados.

# Exemplo de uso do scanf para leitura de strings

```
1  #include<bits/stdc++.h>
2
3  int main(){
4      char palavra[20];
5      printf("Digite uma palavra:\n");
6      scanf("%[^\\n]",palavra);
7      printf("A palavra digitada e: %s\\n",palavra);
8      return 0;
9  }
```



# puts(char \*palavra)

---

- Envia para o dispositivo de saída padrão (tela) o conteúdo do vetor passado como argumento;
- Pode ser usado como alternativa para
  - printf ("%s",palavra)que, no entanto, oferece mais opções de formatação.



# Exemplo de uso do puts

---

- Observe que o puts pode receber somente um parâmetro

```
1  #include<bits/stdc++.h>
2
3  int main(){
4      char palavra[20];
5      puts("Digite uma palavra");
6      scanf("%[^\\n]",palavra);
7      puts("A palavra digitada e: ");
8      puts(palavra);
9      return 0;
10 }
```

# Manipulação de Strings

- Exemplo para realizar a leitura de uma string e imprimí-la mostrando caracter por caracter, um por linha.

```
1  #include<bits/stdc++.h>
2
3  main() {
4      int i;
5      char nome[20];
6      printf("Digite seu nome:\n");
7      gets(nome);
8      for(i=0; nome[i]!='\0'; i++) {
9          printf("%c\n", nome[i]);
10     }
11 }
```

Por exemplo:  
Maria

# Manipulação de Strings

- Exemplo para realizar a leitura de uma string e imprimí-la mostrando caracter por caracter, um por linha.

```
1  #include<bits/stdc++.h>
2
3  main() {
4      int i;
5      char nome[20];
6      printf("Digite seu nome:\n");
7      gets(nome);
8      for(i=0; nome[i]!='\0'; i++) {
9          printf("%c\n", nome[i]);
10     }
11 }
```

→ nome[0]='M'

M



# Manipulação de Strings

- Exemplo para realizar a leitura de uma string e imprimí-la mostrando caracter por caracter, um por linha.

```
1  #include<bits/stdc++.h>
2
3  main() {
4      int i;
5      char nome[20];
6      printf("Digite seu nome:\n");
7      gets(nome);
8      for(i=0; nome[i]!='\0'; i++) {
9          printf("%c\n", nome[i]);
10     }
11 }
```

→ nome[1]='a'

M  
a

# Manipulação de Strings

- Exemplo para realizar a leitura de uma string e imprimí-la mostrando caracter por caracter, um por linha.

```
1  #include<bits/stdc++.h>
2
3  main() {
4      int i;
5      char nome[20];
6      printf("Digite seu nome:\n");
7      gets(nome);
8      for(i=0; nome[i]!='\0'; i++) {
9          printf("%c\n", nome[i]);
10     }
11 }
```

→ nome[2]='r'

M  
a  
r

# Manipulação de Strings

- Exemplo para realizar a leitura de uma string e imprimí-la mostrando caracter por caracter, um por linha.

```
1  #include<bits/stdc++.h>
2
3  main() {
4      int i;
5      char nome[20];
6      printf("Digite seu nome:\n");
7      gets(nome);
8      for(i=0; nome[i]!='\0'; i++) {
9          printf("%c\n", nome[i]);
10     }
11 }
```

nome[3]='i'

M  
a  
r  
i

# Manipulação de Strings

- Exemplo para realizar a leitura de uma string e imprimí-la mostrando caracter por caracter, um por linha.

```
1  #include<bits/stdc++.h>
2
3  main() {
4      int i;
5      char nome[20];
6      printf("Digite seu nome:\n");
7      gets(nome);
8      for(i=0; nome[i]!='\0'; i++) {
9          printf("%c\n", nome[i]);
10     }
11 }
```

→ nome[4]='a'

M  
a  
r  
i  
a

# Manipulação de Strings

- Exemplo para realizar a leitura de uma string e imprimí-la mostrando caracter por caracter, um por linha.

```
1  #include<bits/stdc++.h>
2
3  main() {
4      int i;
5      char nome[20];
6      printf("Digite seu nome:\n");
7      gets(nome);
8      for(i=0; nome[i]!='\0'; i++) {
9          printf("%c\n", nome[i]);
10     }
11 }
```

nome[5]='\0' → pára

M  
a  
r  
i  
a



# Comandos para manipulação de strings

---

- A biblioteca `<bits/stdc++.h>` disponibiliza uma série de operações para manipulação de strings como um objeto, sem ter que indexar seus elementos individualmente.
- Principais funções:
  - `strcpy()`
  - `strlen()`
  - `strcmp()`
  - `strcat()`



# strcpy

---

**char \*strcpy(char \*s1, char \*s2)**

- Copia os caracteres da cadeia s2 para a cadeia s1 e acrescenta o '\0' no final;
- O conteúdo original de s1 é perdido;
- A cadeia s2 permanece inalterada;
- Retorna um ponteiro para s1.



# Exemplo de uso do strcpy

---

```
strcpy(s, "strings");
```

	<i>0</i>	<i>1</i>	<i>2</i>	<i>3</i>	<i>4</i>	<i>5</i>	<i>6</i>	<i>7</i>	<i>8</i>	<i>9</i>
<i>s</i>	's'	't'	'r'	'i'	'n'	'g'	's'	'\0'	?	?





# Exemplo de uso do strcpy

---

```
strcpy(s, "teste");
```

	<i>0</i>	<i>1</i>	<i>2</i>	<i>3</i>	<i>4</i>	<i>5</i>	<i>6</i>	<i>7</i>	<i>8</i>	<i>9</i>
s	't'	'e'	's'	't'	'e'	'\0'	's'	'\0'	'?	'?

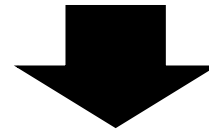


## Exemplo - strcpy

---

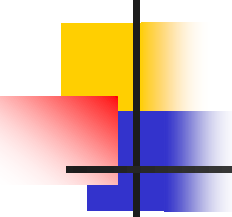
- Faça um programa que armazene duas palavras em duas variáveis e depois realize a troca destas palavras nas variáveis.

Ex.: aluno1="Maria"      aluno2="Jose"



aluno1="Jose"      aluno2="Maria"

*Dica: Utilize uma variável auxiliar*



```
1  #include<bits/stdc++.h>
2
3  int main() {
4      char nome1[20], nome2[20], nomeAux[20];
5      scanf("%s", nome1);
6      scanf("%s", nome2);
7      strcpy(nomeAux, nome1);
8      strcpy(nome1, nome2);
9      strcpy(nome2, nomeAux);
10     printf("nome1 = %s\t nome2 = %s", nome1, nome2);
11     return 0;
12 }
```



# strlen

---

**int strlen(char \*s)**

- Retorna o comprimento (ou seja, a quantidade de caracteres) da cadeia s;



# Exemplo de uso do strlen

---

	<i>0</i>	<i>1</i>	<i>2</i>	<i>3</i>	<i>4</i>	<i>5</i>	<i>6</i>	<i>7</i>	<i>8</i>	<i>9</i>
s	's'	't'	'r'	'i'	'n'	'g'	's'	'\0'	'?	'?

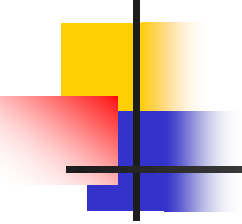
```
int x;  
char s[10]  
x=strlen(s);
```



# Exemplo - strlen

---

- Faça um programa que receba uma string e verifique quantos caracteres a compõe.



---

```
1  #include<bits/stdc++.h>
2
3  int main(){
4      int x;
5      char palavra[20];
6      scanf(" %[^\\n]",palavra);
7      x=strlen(palavra);
8      printf("A palavra %s possui %d caracteres\\n",palavra,x);
9      return 0;
10 }
```



# strcmp

---

**int strcmp(char \*s1, char \*s2)**

- Compara as cadeias s1 e s2 do ponto de vista lexicográfico;
- Retorna:
  - 0 se elas forem idênticas ;
  - um valor qualquer menor que zero se s1 antecede s2;
  - um valor qualquer maior que zero se s1 sucede s2.





# strcmp

---

**int strcmp(char \*s1, char \*s2)**

■ Exemplos:

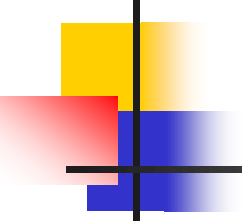
- "ABC" antecede "DEF", portanto `strcmp("ABC","DEF") < 0;`
- "XYZ" é igual a "XYZ", portanto `strcmp("XYZ","XYZ") == 0;`
- "KL" sucede "GDS", portanto `strcmp("KL","GDS") > 0;`
- "KL" antecede "KML", portanto `strcmp("KL","KLM") < 0;`



## Exemplo - strcmp

---

- Faça um programa que receba duas palavras e verifique se elas são iguais.



```
1  #include<bits/stdc++.h>
2
3  int main() {
4      int x;
5      char palavra1[20], palavra2[20];
6      scanf(" %s", palavra1);
7      scanf(" %s", palavra2);
8      if(strcmp(palavra1, palavra2) == 0)
9          printf("As palavras sao iguais\n");
10     else
11         printf("As palavras sao diferentes\n");
12     return 0;
13 }
```



# strcat

---

**char \*strcat(char \*s1, char \*s2)**

- Copia os caracteres da cadeia s2 **para o final** da cadeia s1 e acrescenta o '\0' no final;
- A cadeia s2 permanece inalterada;



# Exemplo de uso do strcat

---

	<i>0</i>	<i>1</i>	<i>2</i>	<i>3</i>	<i>4</i>	<i>5</i>	<i>6</i>	<i>7</i>	<i>8</i>	<i>9</i>
s	'a'	'l'	'f'	'a'	'\0'	'?	'?	'?	'?	'?
t	'b'	'e'	't'	'a'	'\0'	'?	'?	'?	'?	'?

```
strcat(s,t);
```



# Exemplo de uso do strcat

---

	<i>0</i>	<i>1</i>	<i>2</i>	<i>3</i>	<i>4</i>	<i>5</i>	<i>6</i>	<i>7</i>	<i>8</i>	<i>9</i>
s	'a'	'l'	'f'	'a'	'b'	'e'	't'	'a'	'\0'	'?
t	'b'	'e'	't'	'a'	'\0'	'?	'?	'?	'?	'?



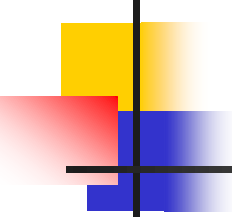
## Exemplo - strcat

---

- Faça um programa que armazene duas palavras em duas variáveis e depois realize a concatenação destas palavras.

Ex.: palavra1="Pseudo" palavra2 ="codigo"

palavra1="Pseudocodigo"



```
1      #include<bits/stdc++.h>
2
3      int main() {
4          int x;
5          char nome1[20], nome2[20];
6          scanf(" %s", nome1);
7          scanf(" %s", nome2);
8          strcat(nome1, nome2);
9          printf("nome1 = %s\n", nome1);
10         printf("nome2 = %s\n", nome2);
11         return 0;
12     }
```





# Comandos para manipulação de caracteres

---

- Principais funções:
  - toupper()
  - tolower()



## char toupper (char letra)

---

- Converte um caracter para maiúsculo.

- Ex.:

```
char palavra[20] = "linguagem";  
palavra[8]=toupper(palavra[8]);  
printf("%s\n",palavra);
```

*Obs.: Será impresso language**M***



## char tolower (char letra)

---

- Converte um caracter para minúsculo.

- Ex.:

```
char palavra[20] = "LINGUAGEM";  
palavra[2]=tolower(palavra[2]);  
printf("%s\n",palavra);
```

*Obs.: Será impresso L**In**GUAGEM*



# Vetores de Strings

---

- Se fizermos um vetor de strings estaremos construindo um vetor de vetores. Esta estrutura é uma matriz bidimensional de char's. Podemos ver a forma geral de uma vetor de strings como sendo:  
**char nome\_da\_variável [num\_de\_strings][compr\_das\_strings];**



# Vetores de Strings

---

- Mas como acessar uma string individual dentro de um laço de repetição?

R.: Usando apenas o primeiro índice.

**nome\_da\_variável [índice]**



# Vetores de Strings

- Ex.: Faça um programa que receba 5 strings, armazene-as em um vetor de strings e exiba-as.

```
1  #include <bits/stdc++.h>
2  int main ()
3  {
4      char strings [5][100];
5      int cont;
6      for (cont=0;cont<5;cont++)
7      {
8          printf ("\n\nDigite uma string: ");
9          scanf(" %s",strings[cont]);
10     }
11
12     printf ("\n\nAs strings que voce digitou foram:\n");
13
14     for (cont=0;cont<5;cont++)
15         printf ("%s\n",strings[cont]);
16
17     return 0;
18 }
```

- 
- 
- Resolver lista de exercícios do URI...