



Caracteres

Arrays de Caracteres (strings)

Caracteres – Char

- ***Char*** é um tipo de dado utilizado para armazenar um único caractere.
- Delimitado por apóstrofos (').

Caracteres – Char

- ***Char*** é um tipo de dado utilizado para armazenar um único caractere.
- Delimitado por apóstrofos (').
- Exemplo de declaração:
char c ;

Caracteres – Char

- ***Char*** é um tipo de dado utilizado para armazenar um único caractere.
- Delimitado por apóstrofos (').
- Exemplo de declaração:
`char c;`
- Exemplos:
`'A' 'a' '1' '?' '+'`
`'\"' '%%' '\\n' '\\t' '\\a'`

- Leitura de um número inteiro e um caractere:

- Leitura de um número inteiro e um caractere:

```
scanf ( "%i" , &num ) ;
```

```
scanf ( "%c" , &letra ) ;
```

- Leitura de um número inteiro e um caractere:

```
scanf( "%i", &num);
```

5 ↵

```
setbuf(stdin, 0); // ou fflush(stdin);
```

```
scanf( "%c", &letra);
```

num = 5

letra = '\n'

- Leitura de um número inteiro e um caractere:

```
scanf( "%i", &num);
```

5 ↵

```
setbuf(stdin, 0); // ou fflush(stdin);
```

```
scanf( "%c", &letra);
```

A ↵

~~num = 5~~

~~letra = '\n'~~

- Leitura de um número inteiro e um caractere:

```
scanf( "%i", &num);
```

5 ↵

```
setbuf(stdin, 0); // ou fflush(stdin);
```

```
scanf( "%c", &letra);
```

A ↵

~~num = 5
letra = '\n'~~

num = 5
letra = 'A'

- Leitura de um número inteiro e um caractere:

```
scanf( "%i", &num);
```

5 ↵

```
setbuf(stdin, 0); // ou fflush(stdin);
```

```
scanf( "%c", &letra);
```

A ↵

~~num = 5
letra = '\n'~~

num = 5
letra = 'A'

- Limpar o *buffer* do arquivo de entrada do teclado.

Arrays de Caracteres – Declaração

- ***String*** é um vetor de caracteres utilizado para armazenar frases e palavras.
- Delimitado por aspas (").

Arrays de Caracteres – Declaração

- ***String*** é um vetor de caracteres utilizado para armazenar frases e palavras.
- Delimitado por aspas (").
- Exemplo de declaração:

```
char texto[80];
```

Arrays de Caracteres – Declaração

- Toda ***String*** é terminada com o caractere ' \0 '. Esse caractere também deve ser considerado na declaração do tamanho da ***string***.

Arrays de Caracteres – Declaração

- Exemplo:

```
char s[ ] = "OLA Mundo" ;
```

Arrays de Caracteres – Declaração

- Exemplo:

```
char s[ ] = "OLA Mundo" ;
```

s[0]	s[1]	s[2]	s[3]	s[4]	s[5]	s[6]	s[7]	s[8]	s[9]
'O'	'L'	'A'	' '	'M'	'u'	'n'	'd'	'o'	'\0'

Arrays de Caracteres – Declaração

- Exemplo:

```
char s[ ] = "OLA Mundo";
```

s[0]	s[1]	s[2]	s[3]	s[4]	s[5]	s[6]	s[7]	s[8]	s[9]
'O'	'L'	'A'	' '	'M'	'u'	'n'	'd'	'o'	'\0'

```
char t[10] = "Piccolo";
```


Arrays de Caracteres – Declaração

- Exemplo:

```
char s[ ] = "OLA Mundo";
```



s[0]	s[1]	s[2]	s[3]	s[4]	s[5]	s[6]	s[7]	s[8]	s[9]
'O'	'L'	'A'	' '	'M'	'u'	'n'	'd'	'o'	'\0'

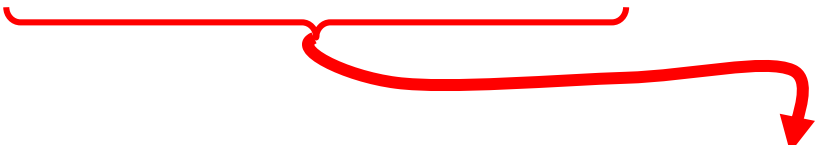
```
char t[10] = "Piccolo";
```

t[0]	t[1]	t[2]	t[3]	t[4]	t[5]	t[6]	t[7]	t[8]	t[9]
'p'	'i'	'c'	'c'	'o'	'l'	'o'	'\0'	'?'	'?'

Arrays de Caracteres – Declaração

- Exemplo:

```
char s[] = "OLA Mundo";
```



s[0]	s[1]	s[2]	s[3]	s[4]	s[5]	s[6]	s[7]	s[8]	s[9]
'O'	'L'	'A'	' '	'M'	'u'	'n'	'd'	'o'	'\0'


```
char t[10] = "Piccolo";
```

t[0]	t[1]	t[2]	t[3]	t[4]	t[5]	t[6]	t[7]	t[8]	t[9]
'P'	'i'	'c'	'c'	'o'	'l'	'o'	'\0'	'?'	'?'

Arrays de Caracteres – Declaração

- Exemplo:

```
char s[ ] = "OLA Mundo";
```



s[0]	s[1]	s[2]	s[3]	s[4]	s[5]	s[6]	s[7]	s[8]	s[9]
'O'	'L'	'A'	' '	'M'	'u'	'n'	'd'	'o'	'\0'

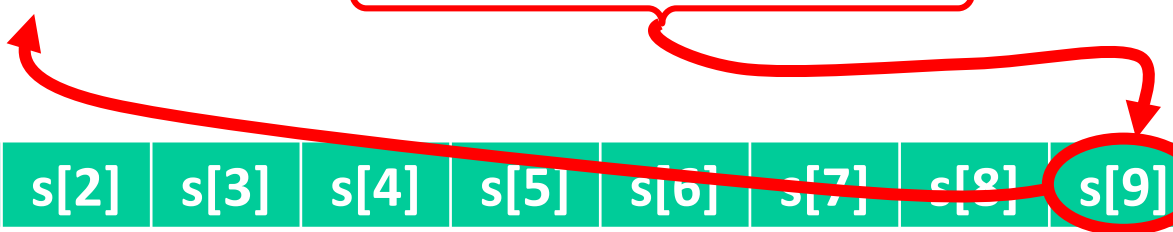
```
char t[10] = "Piccolo";
```

t[0]	t[1]	t[2]	t[3]	t[4]	t[5]	t[6]	t[7]	t[8]	t[9]
'p'	'i'	'c'	'c'	'o'	'l'	'o'	'\0'	'?'	'?'

Arrays de Caracteres – Declaração

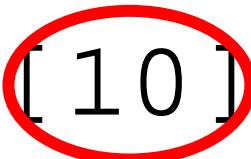
- Exemplo:

```
char s[] = "OLA Mundo";
```



s[0]	s[1]	s[2]	s[3]	s[4]	s[5]	s[6]	s[7]	s[8]	s[9]
'O'	'L'	'A'	' '	'M'	'u'	'n'	'd'	'o'	'\0'

```
char t[10] = "Piccolo";
```

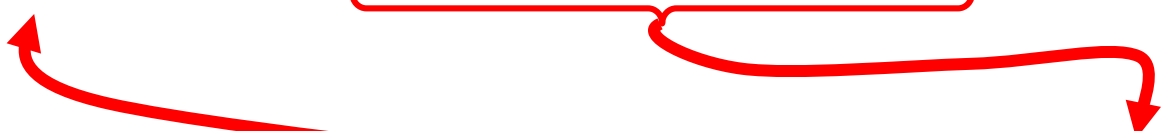


t[0]	t[1]	t[2]	t[3]	t[4]	t[5]	t[6]	t[7]	t[8]	t[9]
'p'	'i'	'c'	'c'	'o'	'l'	'o'	'\0'	'?'	'?'

Arrays de Caracteres – Declaração


- Exemplo:

```
char s[] = "OLA Mundo";
```



s[0]	s[1]	s[2]	s[3]	s[4]	s[5]	s[6]	s[7]	s[8]	s[9]
'O'	'L'	'A'	' '	'M'	'u'	'n'	'd'	'o'	'\0'

```
char t[10] = "Piccolo";
```

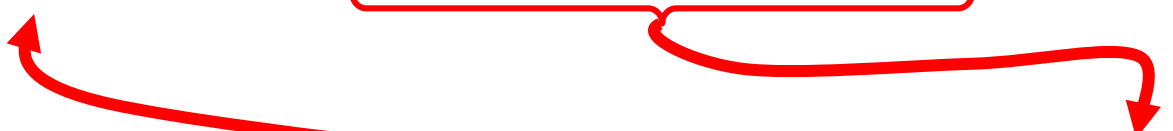


t[0]	t[1]	t[2]	t[3]	t[4]	t[5]	t[6]	t[7]	t[8]	t[9]
'P'	'i'	'c'	'c'	'o'	'l'	'o'	'\0'	'?'	'?'

Arrays de Caracteres – Declaração

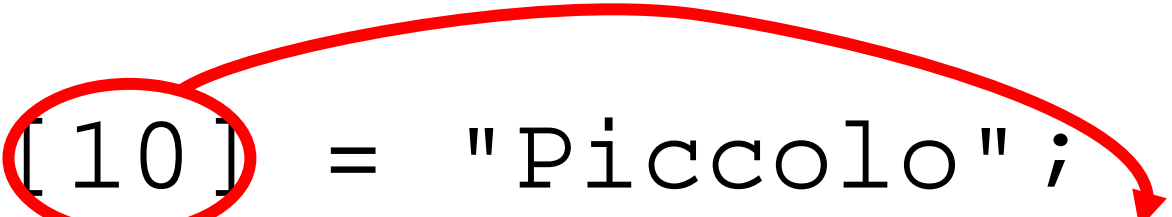
- Exemplo:

```
char s[] = "OLA Mundo";
```



s[0]	s[1]	s[2]	s[3]	s[4]	s[5]	s[6]	s[7]	s[8]	s[9]
'O'	'L'	'A'	' '	'M'	'u'	'n'	'd'	'o'	'\0'

```
char t[10] = "Piccolo";
```



t[0]	t[1]	t[2]	t[3]	t[4]	t[5]	t[6]	t[7]	t[8]	t[9]
'P'	'i'	'c'	'c'	'o'	'l'	'o'	'\0'	'?	'?

Exemplo: declaração de *strings*.

Exemplo: declaração de *strings*.

```
#include <stdio.h>
#define MAX 50

int main( )
{
    char a[MAX] = "Ola Mundo! ";

    printf("Exibe String:%s\n", a);
    printf("Exibe a primeira letra: %c\n", a[0]);

    return 0;
}
```


Exemplo: declaração de *strings*.

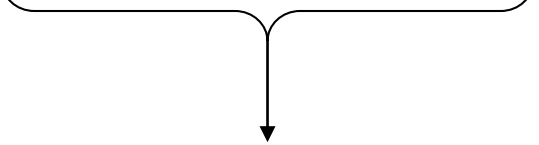
```
#include <stdio.h>
#define MAX 50
```

```
int main( )
{
    char a[MAX] = "Ola Mundo! ";

    printf("Exibe String: %s\n", a);
    printf("Exibe a primeira letra: %c\n", a[0]);

    return 0;
}
```

Acessa um caractere por vez, como um vetor



```
char str1[30];  
scanf( "%s" , str1 );
```

Arrays de Caracteres – Leitura


```
char str1[30];  
scanf("%s", str1);
```



Alguma coisa←

Arrays de Caracteres – Leitura

```
char str1[30];  
scanf("%s", str1);
```




Sem &

Alguma coisa←

str1 = "Alguma"

Arrays de Caracteres – Leitura

```
char str1[30];  
scanf("%s", str1);
```




Sem &

```
Alguma coisa↵
```

str1 = "Alguma"

Arrays de Caracteres – Leitura

```
char str1[30];  
scanf("%s", str1);
```



Sem &

Alguma coisa←

str1 = "Alguma"

Arrays de Caracteres – Leitura

```
char str1[30];
```

```
scanf("%s", str1);
```

 **Sem &**

```
Alguma coisa←
```

str1 = "Alguma"

- Para realizar a ***leitura*** de uma ***string*** digitada pelo usuário, utilizar a função ***fgets()***.

- Para realizar a ***leitura*** de uma ***string*** digitada pelo usuário, utilizar a função ***fgets()***.

```
char str1[15];
```

```
scanf("%i", &num);
```

```
fgets(str1, 15, stdin);
```

- Para realizar a ***leitura*** de uma ***string*** digitada pelo usuário, utilizar a função ***fgets()***.

```
char str1[15];  
scanf("%i", &num);
```

5 ←

```
fgets(str1, 15, stdin);
```

Arrays de Caracteres – Leitura

- Para realizar a ***leitura*** de uma ***string*** digitada pelo usuário, utilizar a função ***fgets()***.

```
char str1[15];  
scanf("%i", &num);
```

5↵

```
fgets(str1, 15, stdin);
```

Um teste↵

Arrays de Caracteres – Leitura

- Para realizar a ***leitura*** de uma ***string*** digitada pelo usuário, utilizar a função ***fgets()***.

```
char str1[15];  
scanf("%i", &num);
```

5 ↵

```
fgets(str1, 15, stdin);
```

Um teste ↵

num = 5

str1 = "\nUm teste\n"

Arrays de Caracteres – Leitura

- Para realizar a ***leitura*** de uma ***string*** digitada pelo usuário, utilizar a função ***fgets()***.

```
char str1[15];  
scanf("%i", &num);
```

5 ↵

```
setbuf(stdin, 0); //fflush(stdin);
```

```
fgets(str1, 15, stdin);
```

Um teste ↵

~~num = 5
str1 = "\nUm teste\n"~~

Arrays de Caracteres – Leitura

- Para realizar a ***leitura*** de uma ***string*** digitada pelo usuário, utilizar a função ***fgets()***.

```
char str1[15];  
scanf("%i", &num);
```

5 ↵

```
setbuf(stdin, 0); //fflush(stdin);
```

```
fgets(str1, 15, stdin);
```

Um teste ↵

~~num = 5
str1 = "\nUm teste\n"~~

num = 5
str1 = "Um teste\n"

Arrays de Caracteres – Leitura

- Para realizar a **leitura** de uma **string** digitada pelo usuário, utilizar a função ***fgets()***.

```
char str1[15];
```

```
scanf("%i", &num);
```

```
setbuf(stdin, 0); //fflush(stdin);
```

```
fgets(str1, 15, stdin);
```

5 ↵

Um teste ↵

~~num = 5~~
~~str1 = "\nUm teste\n"~~

num = 5
str1 = "Um teste\n"

- CUIDADO: Ao utilizar a função **fgets()**, o último caractere da **string** pode ser o **'\n'**.
- Exemplo:

```
char s[10];
```

```
//Usuário digita "Goku"
```

```
fgets(s,10,stdin);
```


Arrays de Caracteres – Leitura

- CUIDADO: Ao utilizar a função **fgets()**, o último caractere da **string** pode ser o **'\n'**.
- Exemplo:

```
char s[10];
```

```
//Usuário digita "Goku"
```

```
fgets(s,10,stdin);
```

s[0]	s[1]	s[2]	s[3]	s[4]	s[5]	s[6]	s[7]	s[8]	s[9]
'G'	'o'	'k'	'u'	'\n'	'\0'	?	?	?	?

Arrays de Caracteres – Leitura

```
char str1[10], str2[10];  
setbuf(stdin, 0);  
fgets(str1, 10, stdin);  
fflush(stdin);  
fgets(str2, 10, stdin);
```

Arrays de Caracteres – Leitura

```
char str1[10], str2[10];  
setbuf(stdin, 0);  
fgets(str1, 10, stdin);  
fflush(stdin);  
fgets(str2, 10, stdin);
```

Alguma coisa←

Arrays de Caracteres – Leitura

```
char str1[10], str2[10];  
setbuf(stdin, 0);  
fgets(str1, 10, stdin);  
fflush(stdin);  
fgets(str2, 10, stdin);
```

Alguma coisa←

str1 = "Alguma co"

Arrays de Caracteres – Leitura

```
char str1[10], str2[10];  
setbuf(stdin, 0);  
fgets(str1, 10, stdin);  
fflush(stdin);  
fgets(str2, 10, stdin);
```

Alguma coisa ←

Algo ←

str1 = "Alguma co"

Arrays de Caracteres – Leitura

```
char str1[10], str2[10];  
setbuf(stdin, 0);  
fgets(str1, 10, stdin);  
fflush(stdin);  
fgets(str2, 10, stdin);
```

Alguma coisa←

Algo←

str1 = "Alguma co" str2 = "Algo\n"

Arrays de Caracteres – Leitura

```
char str1[10], str2[10];  
setbuf(stdin, 0);  
fgets(str1, 10, stdin);  
fflush(stdin);  
fgets(str2, 10, stdin);
```

Alguma coisa←

Algo←

str1 = "Alguma co" str2 = "Algo\n"

O último caractere pode ser o '\n'.

Arrays de Caracteres – Leitura

```
void LerString(char s[])  
{  
    setbuf(stdin, 0);  
    // ou fflush(stdin);  
    fgets(s, MAX, stdin);  
    if (s[strlen(s)-1] == '\\n')  
        s[strlen(s)-1] = '\\0';  
}
```


Arrays de Caracteres – Leitura

```
#define MAX 100
```

```
void LerString(char s[])  
{  
    setbuf(stdin, 0);  
    // ou fflush(stdin);  
    fgets(s, MAX, stdin);  
    if (s[strlen(s)-1] == '\n')  
        s[strlen(s)-1] = '\0';  
}
```

- A linguagem C disponibiliza uma biblioteca para trabalhar com ***strings***.

- A linguagem C disponibiliza uma biblioteca para trabalhar com ***strings***.
- Exemplo de uso da biblioteca:

```
#include <string.h>
```

- ***strlen()*** – retorna a quantidade de caracteres dentro de uma ***string*** antes do caractere '\0'.

Arrays de Caracteres – Tamanho

- ***strlen()*** – retorna a quantidade de caracteres dentro de uma ***string*** antes do caractere '\0'.
- Exemplo:

```
int i;  
char nome[50] = "Jiraya";  
for (i=0; i<strlen(nome); i++)  
    printf( "%c", nome[i] );
```

```
str1 = str2;
```

- ***strcpy(char destino[], char origem[])***
– copia a ***string*** da origem para o destino. A ***string*** destino deve ser longa o bastante.

~~str1 = str2;~~

- ***strcpy(char destino[], char origem[])***
– copia a ***string*** da origem para o destino. A ***string*** destino deve ser longa o bastante.
- Exemplo:

```
char str1[50] = "Jaspion";  
char str2[50];  
strcpy(str2, str1);
```

- ***strcat(char destino[], char origem[])*** – copia a ***string*** da origem para o final da ***string*** destino. A ***string*** destino deve ser longa o bastante.

- ***strcat(char destino[], char origem[])*** – copia a ***string*** da origem para o final da ***string*** destino. A ***string*** destino deve ser longa o bastante.
- Exemplo:

```
char str1[50] = "Power";  
char str2[50] = "Ranger";  
strcat(str1, str2);
```

- ***strcmp(char s1[], char s2[])*** –
Compara s1 com s2. Se s1 for ***igual*** a s2, ***retorna 0***, senão retorna um valor diferente de 0. *Case sensitive.*

- ***strcmp(char s1[], char s2[])*** – Compara s1 com s2. Se s1 for ***igual*** a s2, ***retorna 0***, senão retorna um valor diferente de 0. ***Case sensitive***.
- Exemplo:

```
char str1[50] = "Goku";  
char str2[50] = "Vegeta";  
if (strcmp(str1, str2) != 0)  
    printf("Sayajin");
```

- ***atoi(char s1[])*** – Converte o valor número da ***string*** s1 para um valor ***inteiro***. Quanto nenhum valor válido é encontrado, **retorna** o valor **0**.
- ***atof(char s2[])*** – Converte o valor número da ***string*** s2 para um valor ***real***. Quanto nenhum valor válido é encontrado, **retorna** o valor **0.0**.

Arrays de Caracteres – Conversão

- Exemplo:

```
int main()  
{  
    char s[MAX];  
    int i;  
    float f;  
    printf("Digite um numero inteiro:\n");  
    LerString(s);  
    i = atoi(s);  
    printf("Digite um numero real:\n");  
    LerString(s);  
    f = atof(s);  
    printf("Numero inteiro: %i\n  
           Numero Real: %f\n", i, f);  
    return 0;  
}
```

Arrays de Caracteres – Conversão

• Exemplo:



```
#define MAX 10
```

```
int main()  
{  
    char s[MAX];  
    int i;  
    float f;  
    printf("Digite um numero inteiro:\n");  
    LerString(s);  
    i = atoi(s);  
    printf("Digite um numero real:\n");  
    LerString(s);  
    f = atof(s);  
    printf("Numero inteiro: %i\n  
           Numero Real: %f\n", i, f);  
    return 0;  
}
```

Arrays de Caracteres – Conversão

• Exemplo:

#define MAX 10

```
int main()
{
    char s[MAX];
    int i;
    float f;
    printf("Digite um numero inteiro:\n");
    LerString(s);
    i = atoi(s);
    printf("Digite um numero real:\n");
    LerString(s);
    f = atof(s);
    printf("Numero inteiro: %i\n
           Numero Real: %f\n", i, f);
    return 0;
}
```

void LerString(char s[]);

```
char s[22];
```

```
LerString(s);
```

não cabem muitos cara


```
char s[22];
```

```
LerString(s);
```

não cabem muitos caracteres

```
char s[22];
```

```
LerString(s);
```

não cabem muitos caracteres

```
if ( s1 == s2 )
```

```
char s[22];
```

```
LerString(s);
```

não cabem muitos caracteres

```
if (s1 == s2)
```

```
char s[22];
```

```
LerString(s);
```

não cabem muitos caracteres

```
if (strcmp(s1, s2) != 0)
```

Arrays de Caracteres – Erros comuns

```
s = s1 + s2;
```

```
strcpy(s, s1);
```

```
strcat(s, s2);
```

Arrays de Caracteres – Erros comuns

~~`s = s1 + s2;`~~

`strcpy(s, s1);`
`strcat(s, s2);`

Arrays de Caracteres – Erros comuns

```
"Local: Instituto Mauá de Tecnologia
S = Latitude: -23.6480281
   Longitude: -46.5731710"

strcpy(s, "Local: ");
strcat(s, nomeLocal);
strcat(s, "\nLatitude: ");
strcat(s,
```

Arrays de Caracteres – Erros comuns

```
"Local: Instituto Mauá de Tecnologia
S = Latitude: -23.6480281
   Longitude: -46.5731710"

strcpy(s, "Local: ");
strcat(s, nomeLocal);
strcat(s, "\nLatitude: ");
strcat(s,
```


Arrays de Caracteres – Erros comuns

```
"Local: Instituto Mauá de Tecnologia  
S = Latitude: -23.6480281  
   Longitude: -46.5731710"  
  
strcpy(s, "Local: ");  
strcat(s, nomeLocal);  
strcat(s, "\nLatitude: ");  
strcat(s, ??
```

Arrays de Caracteres – Erros comuns

```
"Local: Instituto Mauá de Tecnologia  
S = Latitude: -23.6480281  
    Longitude: -46.5731710"
```

```
sprintf(s,  
    "Local: %s\n  
    Latitude %.7f\n  
    Longitude: %.7f",  
    nomeLocal, lat, lon);
```

Arrays de Caracteres – Erros comuns

```
"Local: Instituto Mauá de Tecnologia  
S = Latitude: -23.6480281  
    Longitude: -46.5731710"
```

```
sprintf(s,  
    "Local: %s\n  
    Latitude %.7f\n  
    Longitude: %.7f",  
    nomeLocal, lat, lon);
```

Muito útil!!

Não precisa declarar string.h

1. Construir uma função que retire os espaços em branco de uma ***string*** enviada como parâmetro.
2. Construir uma função que realize a inversão de uma ***string*** enviada como parâmetro.

3. Utilizando as funções anteriores, construir um programa que determina se uma ***string*** que o usuário digitou é ou não um palíndromo. Um palíndromo pode ser lido da mesma forma da esquerda para direita, como da direita para esquerda. Exemplos: arara, asa, osso, rever, 121.