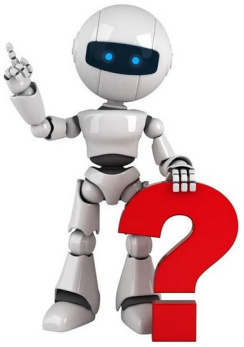
Algoritmos e Estruturas de Dados III

“Cadeias de Caracteres”

Prof. Dr. Felipe Oliveira

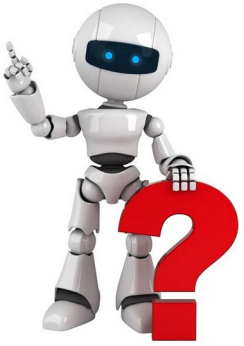
Caractere



- Um caractere é considerado um **tipo de dado primitivo** na maioria dos computadores;
- Um caractere pertence a um conjunto finito de caracteres: um **alfabeto.**

!	@	#	$	%	¨
&	*	( )	+	=	§
{ }	[ ]	?	a	o	^
~	< >	/	´	Ç	º

Caractere

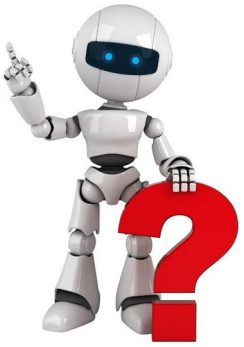


- Os caracteres são representados por **códigos numéricos;**

Como a Linguagem C representa os caracteres

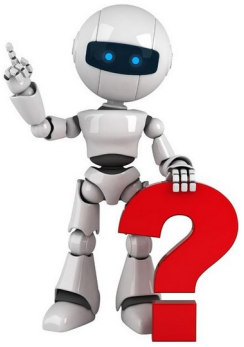
????

Caractere



- Os caracteres são representados por **códigos numéricos;**
- A linguagem C oferece o tipo **char;**
- Um **char** tem tamanho de **1 byte, 8 bits,** e sua versão com sinal pode representar valores que variam de **-128 a 127.**

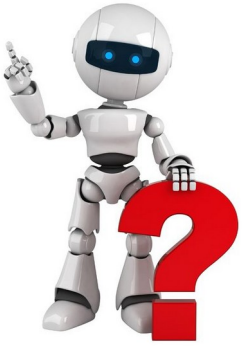
Caractere



- A correspondência entre os caracteres e seus códigos numéricos é feita por uma **tabela de códigos**, chamada **ASCII**.

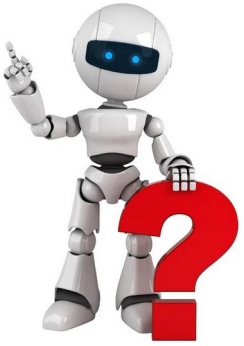
	0	1	2	3	4	5	6	7	8	9
30			sp	!	"	#	$	%	&	'
40	(	)	*	+	,	-	.	/	0	1
50	2	3	4	5	6	7	8	9	:	;
60	<	=	>	?	@	A	B	C	D	E
70	F	G	H	I	J	K	L	M	N	O
80	P	Q	R	S	T	U	V	W	X	Y
90	Z	[	\	]	^	_	`	a	b	c
100	d	e	f	g	h	i	j	k	l	m
110	n	o	p	q	r	S	t	u	v	w
120	x	y	z	{	|	}	~			

Caractere



- Alguns caracteres especiais:
  - '\0' (caractere nulo)
  - '\n' (newline)
  - '\t' (tabulação)
  - '\'' (apóstrofo)
  - '\"' (aspas)

Caractere



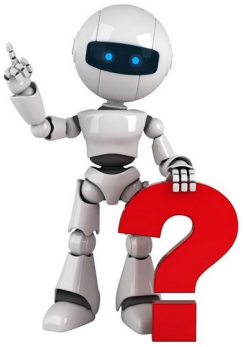
- **Variáveis** do tipo **caractere** são **declaradas** como char:

- **Exemplo:**

  char c1, c2;

  char x=’a’, y=’\n’;

Caractere

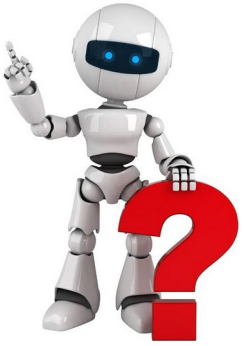


- Em C, a diferença entre **caracteres** e **inteiros** é feita apenas através da maneira pela qual são **tratados.**

- **Exemplo:**

```
char c = 97;
printf("%d %c\n",c,c);
```

Caractere

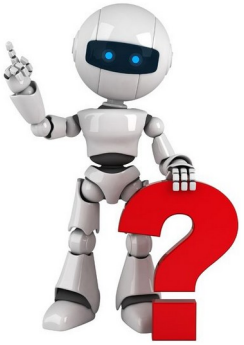


- A linguagem C permite a escrita de **constantes caracteres.**

- **Exemplo:**

```
char c = 'a';
printf("%d %c\n", c, c);
```

Caractere

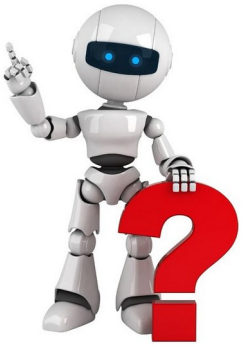


- **função** para **testar** se um **caractere** 'c' é um **dígito** (um dos caracteres entre '0' e '9'):

```
int digito(char c)
{
  if ( (c>='0') && (c<='9') )
    return 1;
  else
    return 0;
}
```

Exercícios

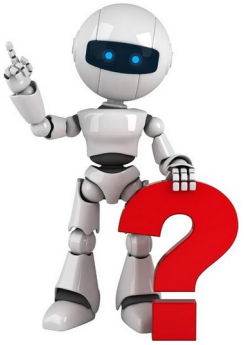
Exercícios - Caractere



- Escreva uma função para **determinar se** um **caractere é** uma **letra**, com protótipo:

```
int letra(char c);
```

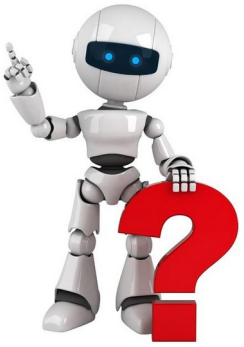
Exercícios - Caractere



- Escreva uma função para **determinar se** um **caractere é** uma **letra**, com protótipo:

```
int letra(char c)
{
   if ( (c>='a') && (c<='z') || (c>='A') && (c<='Z') )
      return 1;
   else
      return 0;
}
```

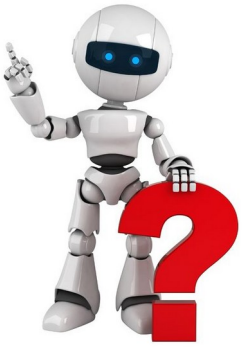
Exercícios - Caractere



- Escreva uma função para **converter** um **caractere para maiúscula**. Se o caractere dado representar uma letra **minúscula**, devemos ter como valor de **retorno** a **letra maiúscula** correspondente. Se o caractere dado não for uma letra minúscula, devemos ter como valor de retorno o mesmo caractere, sem alteração.
- O protótipo desta função pode ser dado por:

- char maiuscula(char c);

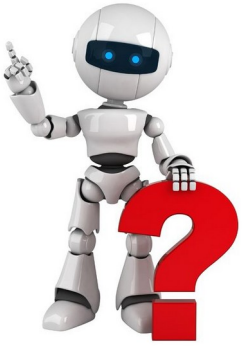
Exercícios - Caractere

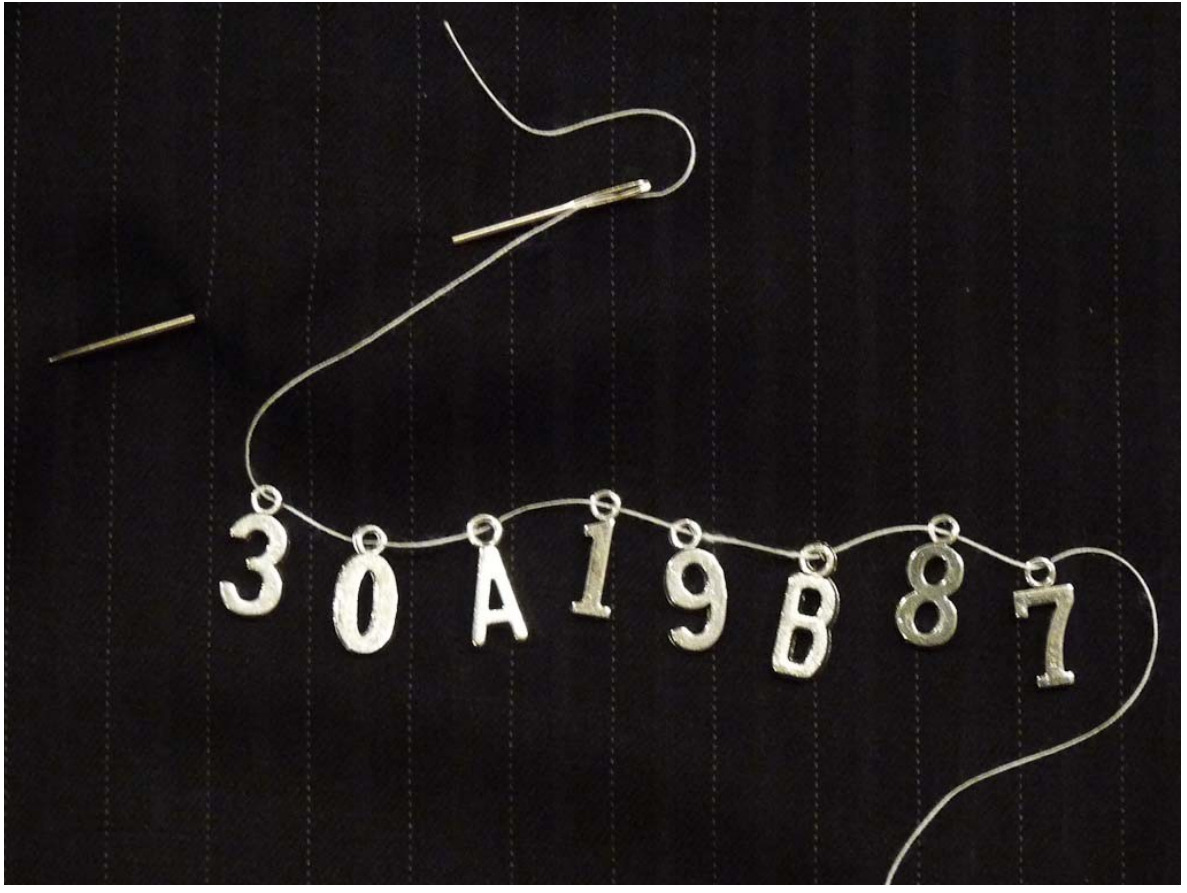


- Escreva uma função para **converter** um **caractere para maiúscula.**

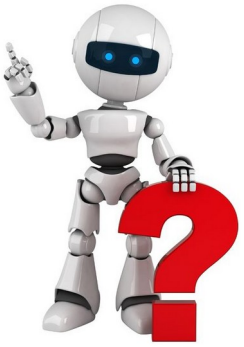
```
char maiuscula(char c)
{
   if ( (c>='a') && (c<='z'))
      return(c-32);
   else if( (c>='A') && (c<='Z') )
      return(c);
}
```

Cadeia de Caracteres



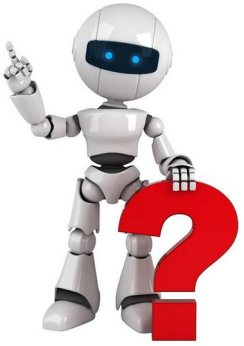


Cadeia de Caracteres



- **Sequência de** elementos denominados Caracteres;
- **Sequência** de **letras** e **símbolos**, onde os símbolos podem ser **espaços** em branco, **dígitos** e vários outros como **pontos de exclamação** e **interrogação**, **símbolos matemáticos**, etc.

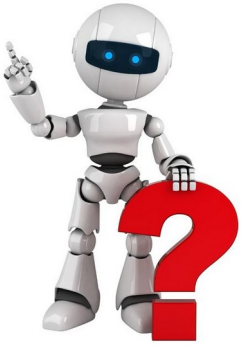
Cadeia de Caracteres



- Em C, uma **cadeia de caracteres** é representada por um **vetor de** variáveis do tipo **char** e é **terminada** com o marcador **'\0';**

- **Cadeias de caracteres** são popularmente conhecidas como **String.**

Cadeia de Caracteres



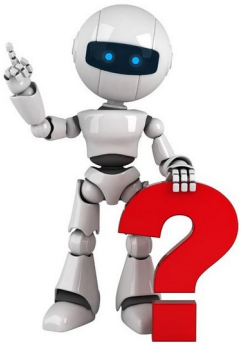
- **Declaração** de uma **cadeia de caracteres:**

- **Exemplo:**

```
char nome[6];
```

Lembrar que o tamanho da string deve também contar o '\0' final (6 = 5 caracteres + '\0' final)

Cadeia de Caracteres

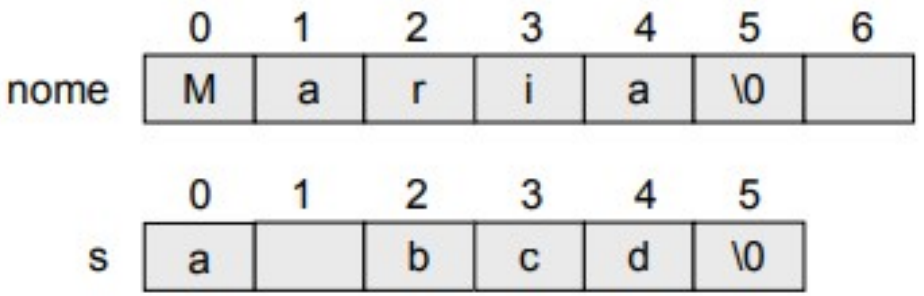


- É possível **declarar** uma **string** e **dar** o seu **valor inicial** junto ao comando de declaração.

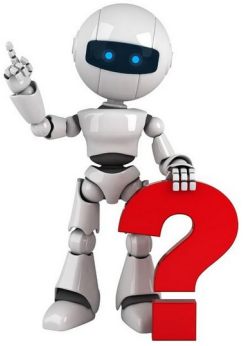
- **Exemplo:**

  char nome[7] = “Maria”;

  char s[6] = “a bcde”;



Cadeia de Caracteres



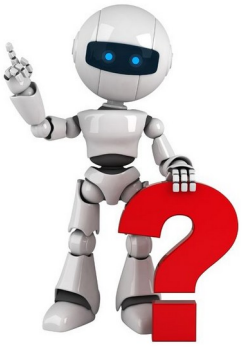
- A sequência especial **%s** deve ser usada nos comando **printf** e **scanf** para **mostrar** ou **ler** uma string, respectivamente.

- **Exemplo:**

  printf(“%s”, texto);

  scanf(“%s”, texto);

Cadeia de Caracteres

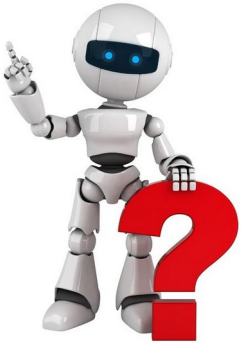


- **Problema!!!**

```
char x[10], y[10];
x = "Ola";
x = y;                  /* Não faça isso! */
x = x + y;              /* Não faça isso! */
if (x == y)...          /* Não faça isso! */
```

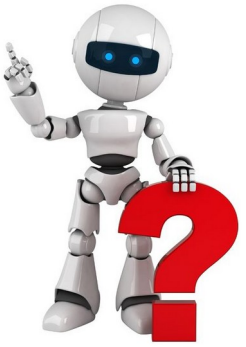
- **Strings não podem** ser **atribuídas** ou **comparadas** diretamente;
- **Strings** são normalmente **manipuladas** por intermédio de funções de **biblioteca:**
  - **#include<string.h>**

Cadeia de Caracteres



- **Strings** são normalmente **manipuladas** por meio de funções da **biblioteca** **#include <string.h>**:
- **strcpy(string_destino,string_origem):**
  - Copia a string-origem para a string-destino.
- **strcat(string_destino,string_origem):**
  - A string de origem permanecerá inalterada e será anexada ao fim da string de destino.
- **strlen(string):**
  - Retorna o comprimento da string fornecida.

Cadeia de Caracteres

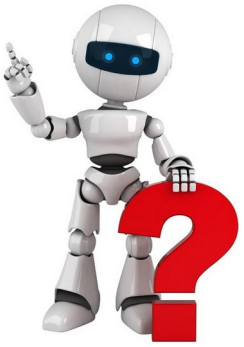


- **Strings** são normalmente **manipuladas** por meio de funções da **biblioteca** **#include <string.h>**:

- **strcmp(string1,string2):**
  - Compara a string1 com a string2, retorna -1 se string1 < string2, zero se string1 == string2 e +1 se string1 > string2. O compilador diferencia maiúsculas de minúsculas bem como de letras com e sem acento.

- **stricmp(string1,string2):**
  - Compara a string1 com a string2, retorna -1 se string1 < string2, zero se string1 == string2 e +1 se string1 > string2. O compilador não diferencia maiúsculas de minúsculas mas diferencia letras com e sem acento.

Cadeia de Caracteres



- **Solução!!!**

  char x[10], y[10];

- **No lugar de:**

  x = “Ola”;

  x = y;

  x = x + y;

  if(x == y)...

  if(x < y)...

  if(x > y)...

Utilize:

strcpy(x,”Ola”);

strcpy(x,y);

strcat(x,y);

if(strcmp(x,y) == 0)...

if(strcmp(x,y) == –1)...

if(strcmp(x,y) == 1)...

Exercícios

Exercícios – Cadeias de Caracteres

- Escreva funções em C para reproduzir os comportamentos das funçoẽs abaixo:
  - strcpy;
  - strcat;
  - strlen;
  - strcmp;

Não utilizar as funções prontas

Exercícios – Cadeias de Caracteres

- Existe uma função na biblioteca string.h chamada **strchr** que acha a primeira ocorrência de um caractere em uma string. Escreva uma implementação para esta função através da manipulação de strings como vetores.

Não utilizar as funções prontas

Exercícios – Cadeias de Caracteres

- Uma cadeia de caracteres é dita ser palíndromo se a sequência dos caracteres da cadeia da esquerda para direita é igual a sequência de caracteres da direita para esquerda.
- **Exemplo:**
  - ABC12321CBA, ACCA, XYZ6.6ZYX.
- Faça uma função que retorna verdadeiro se a cadeia de caracteres enviada como parâmetro é palíndromo.

Não utilizar as funções prontas

Exercícios – Cadeias de Caracteres

- Escreva uma função em C que receba uma string e exiba uma tabela com o número de ocorrências de cada caractere na string. Escreva um programa para testar sua função.

Não utilizar as funções prontas

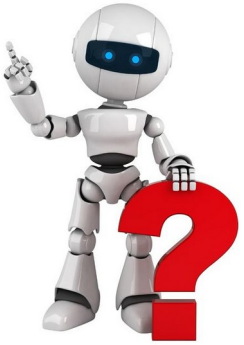
Exercícios – Cadeias de Caracteres

- Escreva uma função em C que imprima uma string, caractere por caractere. Escreva um programa para testar sua função.

Não utilizar as funções prontas

Dúvidas

Atividade Prática



- Implemente os exercícios que não foram resolvidos em sala de aula;
- Revise os exercícios resolvidos.

Algoritmos e Estruturas de Dados III

Prof. Dr. Felipe Oliveira
felipeoliveira@ufam.edu.br