Algoritmia e

Programação

Cadeias de caracteres (strings).



Texto: uma sequência de caracteres

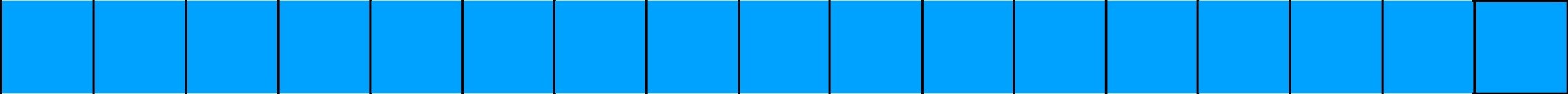
•

•

Uma palavra ou um texto são um conjunto de caracteres.

A linguagem de programação C não tem um tipo específico para texto.

* Uma cadeia de caracteres (*string*) é uma **sequência de char**’s terminada pelo carácter nulo '**\0**'.



U

m

a

s

t

r

i

n

g

e

m

C

.



\0

**APROG - LETI**

Declarar uma string

•

•

As strings são armazenadas em vectores do tipo char.

A dimensão de um vector deve ser suficiente para a maior string prevista, incluindo o carácter '\0'.



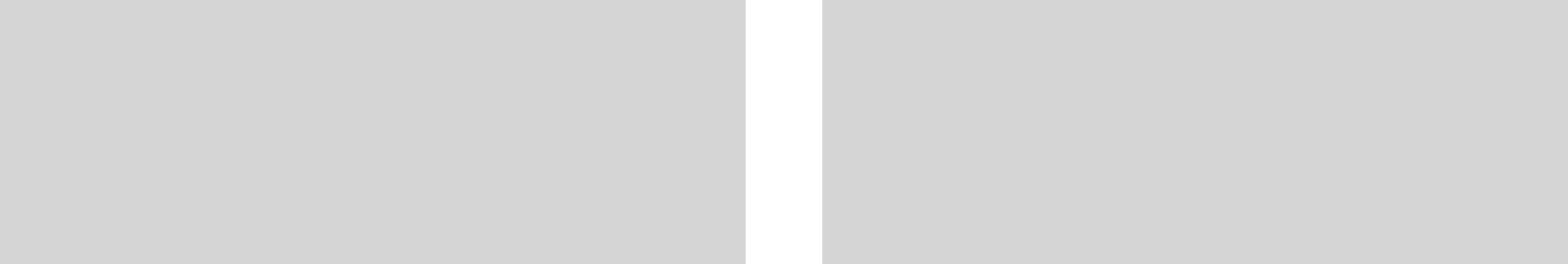
* Capacidade para 49 caracteres úteis + '\0': char s1[50];
* Vector com dimensão 12 ('\0' incluído): char s2[] = "Uma string.";
* Vector com dimensão 50 (14 elementos ocupados). char s3[50] = "Outra string.";



**APROG - LETI**

Ler uma string

* É possível ler uma string com a função scanf()…
* … mas esta função só lê texto até ao primeiro espaço branco.



char s[80];

**printf**("Nome: ");

**scanf**("%s", s);

**printf**(Olá, %s.\n", s);

*<-- RUN -->*

Nome: Afonso Henriques Olá, Afonso.



**APROG - LETI**

Ler uma string

* Para ler uma linha de texto, utiliza-se a função fgets() com os parâmetros:
  + vector para guardar a string,

char s[80];



**printf**("Nome: ");

**fgets**(s, sizeof(s), stdin);

**printf**(Olá, %s.\n", s);

•

•

|  |  |  |
| --- | --- | --- |
| número máximo de |  |  |
| *<-- RUN -->* |  |
| caracteres a ler, | Nome: Afonso Henriques |  |
| origem da string. | Olá, Afonso Henriques. |  |
|  |  |
|  |  |  |



**APROG - LETI**

scanf e fgets : eterno problema

•

•

A função fgets() lê da origem de dados todos os caracteres até encontrar o carácter '\n' (tecla ENTER).

A função scanf() lê (retira) da origem de dados apenas os caracteres que correspondem ao formato especificado.

* Uma função scanf() executada antes de uma função fgets() pode deixar caracteres na origem de dados que serão incorrectamente assumidos como os caracteres a ler!



**APROG - LETI**

scanf() e fgets() : eterno problema

* Exemplo:



int n;

char p[80];

**printf**("Número: "); **scanf**("%d", &n);

**printf**("String: ");

**fgets**(p, sizeof(p), stdin);

**printf**("Lido: %s.\n", p);

*<-- RUN -->*



Número: 14

String:

Lido: .

**Digitado:** '1' + '4' + <ENTER>

*Antes do scanf()*

stdin: "14\n"

|  |  |  |
| --- | --- | --- |
| *Após o scanf()* | É necessário “limpar” |  |
| o stdin antes de |  |
| stdin: "\n" |  |
| executar o fgets()! |  |
|  |  |



**APROG - LETI**

scanf() e fgets() : possível solução

* Exemplo:

int n;

char p[80];

**printf**("Número: ");

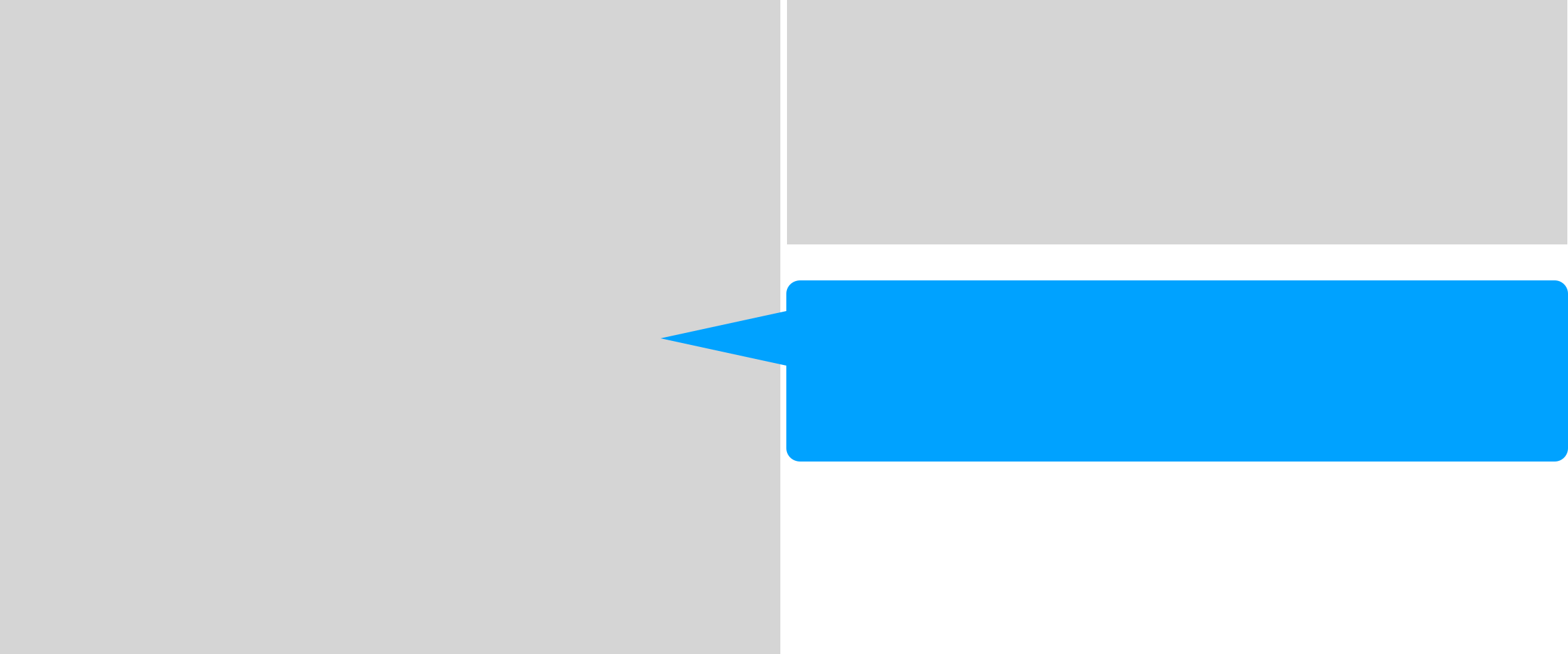
**scanf**("%d", &n);

**while**(getchar() != '\n');

**printf**("String: ");

**fgets**(p, sizeof(p), stdin);

*<-- RUN -->*



Número: 14

String: Reactor nuclear

Lido: Reactor nuclear.

**Este ciclo retira todos os caracteres no stdin. Quando o fgets for executado, o stdin estará vazio.**

**printf**("Lido: %s.\n", p);



**APROG - LETI**

Operações com strings

* As strings são armazenadas em vectores.
  + Não é possível executar as operações básicas da mesma forma que é possível com char, int ou float.
* As operações sobre strings são realizadas operando sobre cada elemento do vector (tal como com vectores de int, por exemplo).



**APROG - LETI**

#include <string.h>

* Esta biblioteca define várias funções úteis para manipular strings.
* Vamos ver:
  + strlen(), memset()
  + strncpy(), strncat()
  + strcmp()
  + strchr(), strstr()
* Para documentação completa, consultar:
  + <http://www.cplusplus.com/reference/cstring/>



**APROG - LETI**

strlen() : comprimento da string

* A função strlen() retorna o comprimento de uma string, excluindo o carácter nulo.



char s[50] = "Uma string.";

int comprimento;

comprimento = strlen(s); *// comprimento = 11*

printf("Comprimento: %d\n", comprimento);



**APROG - LETI**

memset() : preencher string

* A função memset() preenche uma string com um determinado carácter.



char s[50];

* *Preencher s com caracteres nulos ('\0')* memset(s, 0, sizeof(s));



**APROG - LETI**

strncpy() : copiar uma string

* A função strncpy() copia uma string para outra string.
* Previne copiar mais caracteres do que a dimensão da string destino.



char destino[50];

char origem[50];

strncpy(origem, "Eu sou uma string!", sizeof(origem));

* *Copiar para destino a string origem.*

strncpy(destino, origem, sizeof(destino));

**APROG - LETI**

strncat() : concatenar duas strings

* A função strncat() acrescenta uma string no final de outra string.



char str1[50] = "Uma string";

char str2[50] = " e mais outra string!"; int folga;

* *Determinar espaço disponível em str1.* folga = sizeof(str1) - (strlen(str1) + 1);
* *Copiar para str1 (destino) a string str2.* strncat(str1, str2, folga);



**APROG - LETI**

strcmp() : comparar duas strings

* A função strcmp() compara duas strings, retornando
  + 0 se forem iguais,
  + valor negativo se a primeira string for alfabeticamente anterior à segunda,
  + valor positivo se a primeira string for alfabeticamente posterior à segunda.



char str1[50] = "Uma string";

char str2[50] = "Uma string";

* *Comparar strings.* if(strcmp(str1, str2) == 0)

printf("São iguais!\n");

**APROG - LETI**

strchr() : encontrar um carácter numa string

* A função strchr() procura um determinado carácter numa string, retornando
  + o endereço de memória onde se encontra a primeira ocorrência do carácter na string, ou
  + o valor *NULL*, se não existir o carácter na string.



char str[50] = "Era uma vez um reino distante..."; char \*pchar;

* *Procurar o carácter na string.* pchar = strchr(str, 'z'); if(pchar != NULL)

printf("O primeiro 'z' está na posição %d.\n", pchar - str); else

printf("A string não tem um 'z'.\n");



**APROG - LETI**

strstr() : encontrar uma substring numa string

* A função strstr() procura uma determinada substring dentro de uma string, retornando
  + o endereço de memória onde se encontra a primeira ocorrência da substring na string, ou
  + o valor *NULL*, se não existir.



char str[50] = "Era uma vez um reino distante..."; char \*pstring;

* *Procurar a substring na string.* pstring = strstr(str, "reino"); if(pstring != NULL)

printf("O primeiro \"reino\" está na posição %d.\n", pstring - str); else

printf("A string não tem \"reino\".\n");

**APROG - LETI**