Fundamentos da Programação

Strings

P.PORTO
ESCOLA SUPERIOR DE TECNOLOGIA E GESTÃO

Conteúdo

- *Array* de caracteres (*String*)
- *Input/output*

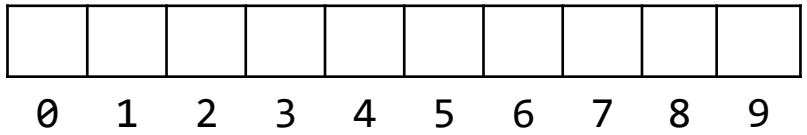
String

- Em C não existe o tipo *string*, sendo implementado através de um *array* de caracteres.
  - Uma *string* é terminada pelo carácter especial **'\0'** indicando o fim da *string*.
    - Este caracter deve ser contabilizado na declaração do tamanho do *array*.

Declaração

- Sintaxe: `char nome_string[nº_de_caracteres]`

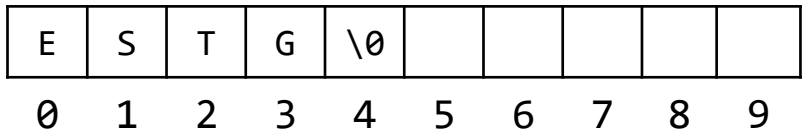
Contabilizando o `'\0'`

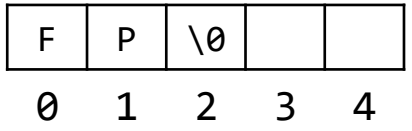



```
…
char palavra[10];
…
```

Inicialização

- `char nome_string[numero_caracteres] = “string”`
- `char nome_string[numero_caracteres] = { caracteres, ‘\0’ }`






```
…
char escola[10] = “ESTG”;
char uc[5] = {‘F’, ‘P’, ‘\0’};
…
```

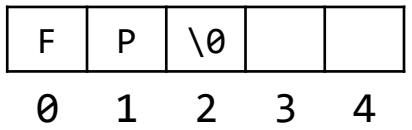
Manipulação

- Não é possível atribuir uma *string* a um *array* de caracteres após a sua inicialização.
  - Isto aplica-se a *arrays* de qualquer tipo em C.
- Neste caso é necessário copiar elemento a elemento (ou utilizar funções específicas para esse efeito).

```
…
char abcd[5];
abcd = “abcd”; // erro
…
```

printf(…)
puts(…)

- `printf(…)`
- `puts(…)`
  - A função **`puts`** escreve a *string* e faz a mudança de linha.




```
…
char uc[5] = {‘F’, ‘P’, ‘\0’};
printf(“A UC é %c %c”, uc[0], uc[1]);
printf(“A UC é %s”, uc);
puts(uc);
…
```

Output

- É fundamental que a *string* esteja terminada com **‘\0’**.

A	B	C	D	E
0	1	2	3	4

```
…
char abcde[5] = {'A', 'B', 'C', 'D', 'E'};
printf("A UC2 é %s", abcde);
…
```

```
A UC2 é ABCDE□□□□
RUN SUCCESSFUL (total time: 102ms)
```

scanf(…)

- `scanf(“%s”, nome_string);`
  - A leitura de caracteres termina quando é encontrado um **espaço**, TAB ou ENTER. Ou seja, só é lida uma palavra.

```
 …
char nome[25];
puts("insira o seu nome");
scanf("%s", nome);
puts(nome);
…
```

`scanf(…)`

- Não se coloca o **&** na leitura de uma strings.
  - A nome da variável, representa o *array*, referencia o endereço de memória para o primeiro elemento, ou seja **`nome_string`** é equivalente a **`&nome_string[0]`**.

```
 …
char nome[25];
puts("insira o seu nome");
scanf("%s", &nome[0]);
puts(nome);
…
```

fgets(…)

- Sintaxe: **`char *fgets(char *str, int num, FILE *stream);`**
  - **`str`**: *string* para onde a sequência de caracteres lida será copiada.
  - **`num`**: número máximo de caracteres a serem copiados para **`str`** (incluindo o **`'\0'`**).
  - **`stream`**: Apontador para um objeto **`FILE`** que identifica um fluxo de entrada.
    - **`stdin`** pode ser usado como argumento para ler a partir da entrada padrão.

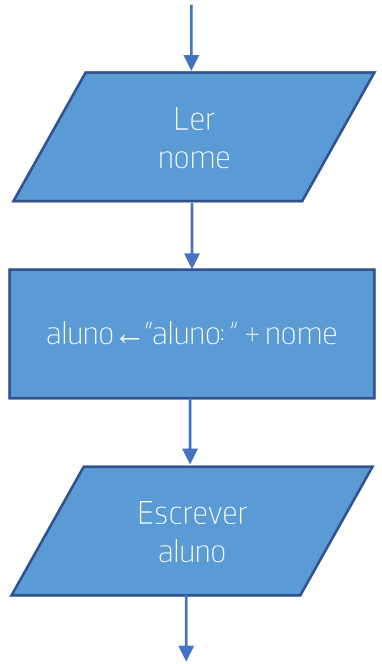
```
…
char nome[30];
printf("Enter nome : ");
fgets(nome, sizeof(nome), stdin);
puts(nome);
…
```

`fgets(…)`

- `fgets()` lê os dados do buffer incluindo `'\n'` e termina a *string* com `'\0'`.
  - Se não existir espaço suficiente no buffer antes de se deparar com a nova linha (`'\0'`), copia o que pode (o comprimento do buffer menos um byte) e termina a linha. A leitura é retomada na próxima iteração a partir de onde o `fgets()` parou.

Strings

```
…
ler nome
aluno ← “aluno: ” + nome
escrever aluno
…
```



Leitura recomendada

- (Capítulo 7) Damas, L. Linguagem C; FCA – Editora de Informática, Lda, 1999; ISBN 9789727221561.