Strings

(cadeia de carateres)

Strings

Definição

- Uma string é um tipo de dados composto definido através de um array de 1 dimensão de carateres

Exemplos

- Literais ou constantes (definidos entre aspas)
  - “Programacao em linguagem C.”
  - “Carlos”
  - “Rua da Pescada, 5; 1100-115 Lisboa”
  - “213445232”
  - “A”

Carateres vs. Strings

Exemplo

- Caráter: 'A'
  - ocupa 1 byte
- String: “A”
  - ocupa 2 bytes
- Conclusão: 'A' ≠ “A”

Caráter terminador de strings

Problema

- Onde termina a string, uma vez que o tamanho do array de carateres que contém a string pode ser superior ao número de carateres da string?

Solução

- Utilizar um caráter terminador para indicar o final da string
- O caráter usado é o primeiro caráter da tabela ASCII: **'\0'**

Exemplo

- Declaração de um array para receber uma string com 20 carateres úteis

```
char  st[21];      // 20 carateres para a string + 1 caráter para o terminador
```

Declaração e iniciação de strings

Exemplo 1

- Tamanho predefinido do array

```
char  s[20] = "PROGRAMACAO";      // 11 carateres + 1 caráter terminador ('\0')
char  s[20] = { 'P', 'R', 'O', 'G', 'R', 'A', 'M', 'A', 'C', 'A', 'O' };      // 11 + 1 carateres
```

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19
s	P	R	O	G	R	A	M	A	C	A	O	\0								

Exemplo 2

- Tamanho indefinido do array

```
char  s[] = "PROGRAMACAO";       // 11 carateres + 1 caráter terminador ('\0')
char  *s = "PROGRAMACAO";        // 11 carateres + 1 caráter terminador ('\0')
```

	0	1	2	3	4	5	6	7	8	9	10	11
s	P	R	O	G	R	A	M	A	C	A	O	\0

Atenção

- A variável **s** é um array que contém o endereço do primeiro elemento do array
- Portanto, é **errado** usar a instrução de atribuição entre strings, exceto na **iniciação**

Arrays vs. Strings

Atenção

- Uma string é um array de carateres com um caráter especial como terminador, o caráter **'\0'**
- Nem todo o array de carateres é uma string

Exemplo

- **String** (com tamanho indefinido do array)

```
char  s[] = "PROGRAMACAO";
char  *s = "PROGRAMACAO";
```

Contra-exemplo

- Array de carateres com tamanho indefinido (**não é string**)

```
char  s[] = { 'P', 'R', 'O', 'G', 'R', 'A', 'M', 'A', 'C', 'A', 'O' };
char  *s = { 'P', 'R', 'O', 'G', 'R', 'A', 'M', 'A', 'C', 'A', 'O' };
```

Escrita e leitura de strings

Escrita no ficheiro padrão (monitor)

- Subprogramas
  - **printf**
  - **puts**
- **puts** (sintaxe)

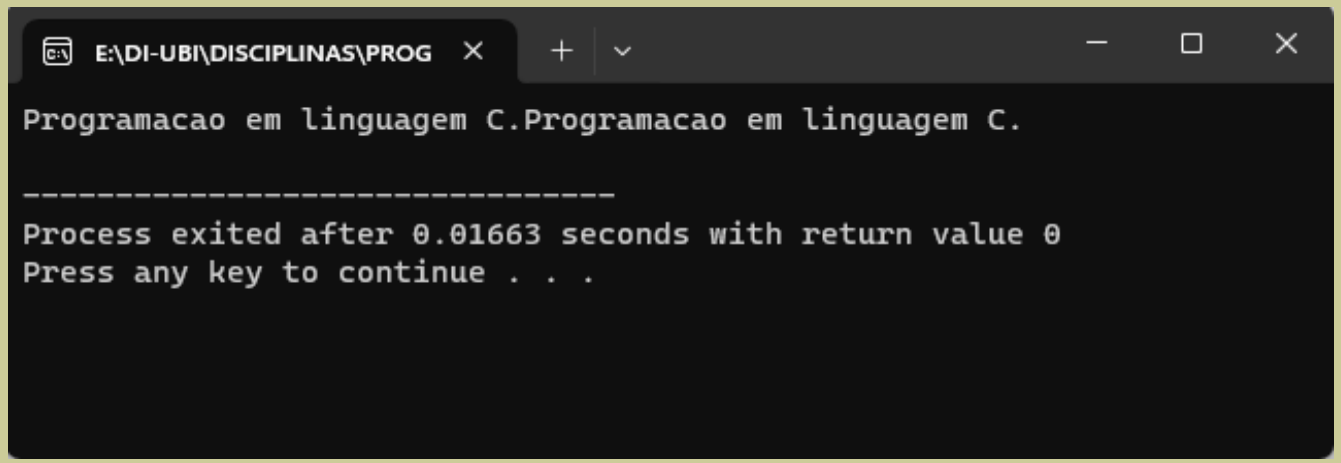
```
int puts (const char *str);
```

  - escreve a string apontada por **str** no ficheiro padrão, mas não escreve o caráter **'\0'**, e acrescenta '**\n'** no fim
  - devolve um valor não negativo (se teve sucesso) ou EOF se houve algum erro
- Notas
  - **puts** é equivalente a **printf + '\n'** (mas só na saída/escrita de uma string)

Escrita no ficheiro padrão (monitor)

- Exemplo 1

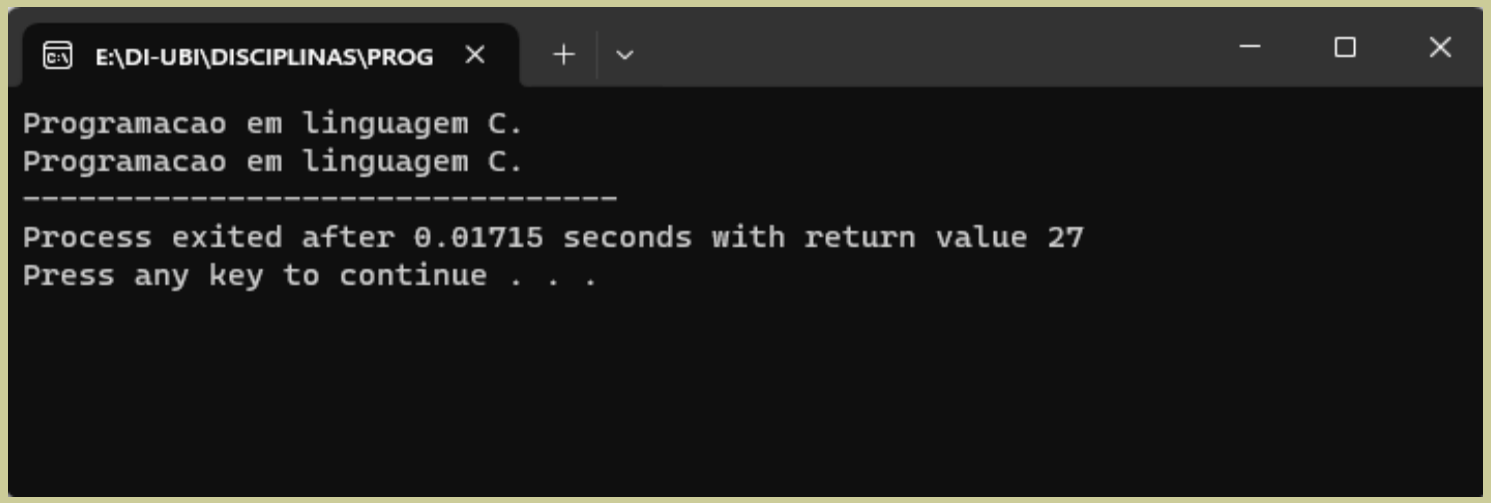
```
#include <stdio.h>
main()
{
  printf("Programação em linguagem C.");
  puts("Programação em linguagem C.");
}
```



Escrita no ficheiro padrão (monitor)

- Exemplo 2

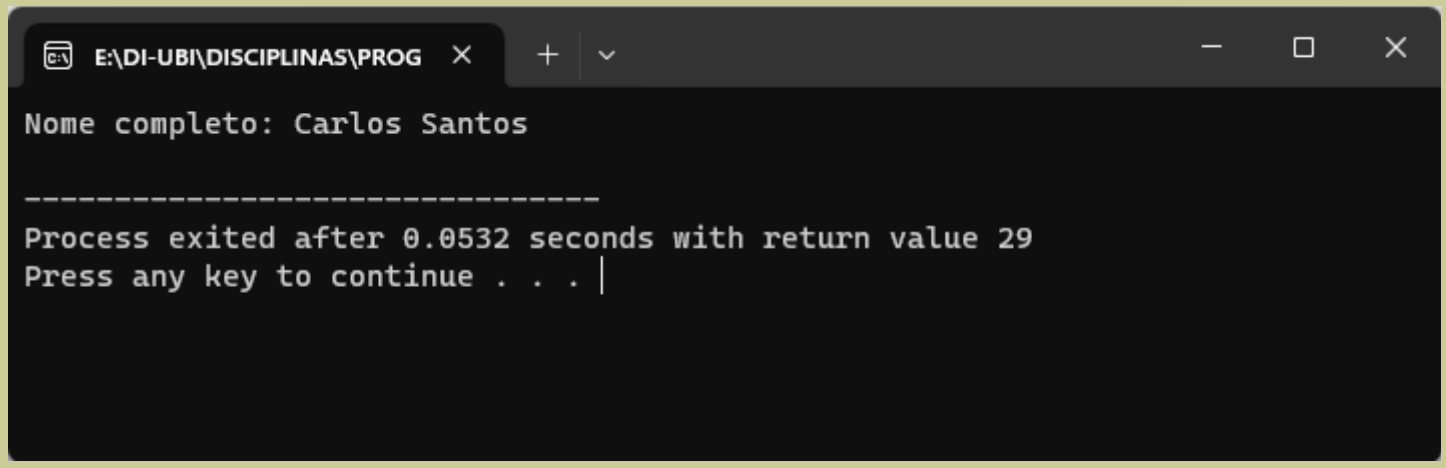
```
#include <stdio.h>
main()
{
  puts("Programação em linguagem C.");
  printf("Programação em linguagem C.");
}
```



Escrita no ficheiro padrão (monitor)

- Exemplo 3

```
#include <stdio.h>
main()
{
  char  nome[20] = "Carlos";
  char apelido[20] = "Santos";
  printf("Nome completo: %s %s\n", nome, apelido);
}
```



Leitura do ficheiro padrão (teclado)

- Subprogramas
  - **scanf**
  - **gets**
- **gets** (sintaxe)

```
char *gets (char *str);
```

  - lê uma linha do ficheiro padrão e guarda-a na string apontada por **str**;
    termina quando ler o caráter '\n' ou o caráter fim-de-ficheiro (o que ocorrer primeiro)
  - devolve o ponteiro **str** (se teve sucesso) ou **NULL** (se houve algum erro na leitura ou se não foi lido qualquer caráter)
- Notas
  - **scanf** não usa o **operador &** na leitura de strings
  - **scanf** só permite ler **uma palavra** (e não uma frase)
  - **gets** lê strings com uma ou várias palavras

Leitura do ficheiro padrão (teclado)

- Exemplo 1 (usando o subprograma **scanf**)

```
#include <stdio.h>
main()
{
  char  nome[20], apelido[20];
  printf("Inserir o nome: ");
  scanf("%s", nome);
  printf("Inserir o apelido: ");
  scanf("%s", apelido);
  printf("Nome completo: %s  %s\n", nome, apelido);
}
```

Leitura do ficheiro padrão (teclado)

- Exemplo 2 (usando o subprograma **gets**)

```
#include <stdio.h>
main()
{
  char  nomeCompleto[40];
  printf("Inserir o nome completo: ");
  gets(nomeCompleto);
  printf("Nome completo: %s\n", nomeCompleto);
}
```

Escrita em ficheiro de texto

- Subprogramas
  - **fprintf**
  - **fputs**
- **fputs** (sintaxe)

```
int fputs (const char *str, FILE *fich);
```

  - escreve a string apontada por **str** no ficheiro **fich**, mas não escreve o caráter **'\0'**, e acrescenta **'\n'** no fim
  - devolve um valor não negativo (se teve sucesso) ou EOF se houve algum erro
- Nota
  - **fputs** é equivalente a **fprintf + '\n'** (mas só na saída/escrita de uma string)

Escrita em ficheiro de texto

- Exemplo

```
#include <stdio.h>
main()
{
  char  nomeCompleto[40] = "Carlos Santos";
  FILE  *f;
  f = fopen("Nome.txt", "w");
  fprintf(f, "%s\n", nomeCompleto);
  fputs(nomeCompleto, f);
  fclose(f);
}
```

Leitura de ficheiro de texto

- Subprogramas
  - **fscanf**
  - **fgets**
- **fgets** (sintaxe)

```
char *fgets (char *str, int n, FILE *fich);
```

  - lê uma linha do ficheiro de texto **fich** e guarda-a na string apontada por **str** (str recebe no máximo **n** carateres, incluindo o caráter '\0'); termina quando forem lidos (n-1) carateres, ou o caráter '\n', ou o caráter fim-de-ficheiro (o que ocorrer primeiro)
  - devolve o ponteiro **str** (se teve sucesso) ou NULL (se houve algum erro na leitura ou se não foi lido qualquer caráter)
- Notas
  - **fscanf** não usa o **operador &** na leitura de strings
  - **fscanf** só permite ler **uma palavra** (e não uma frase)
  - **fgets** lê strings com uma ou mais palavras

Leitura de ficheiro de texto

- Exemplo:

```
#include <stdio.h>
main()
{
  char  nome[40];
  FILE  *f;
  f = fopen("Nomes.txt", "r");
  fscanf(f, "%s", nome);
  printf("Nome: %s\n", nome);
  fgets(nome, 40, f);
  printf("Nome: %s\n", nome);
  fclose(f);
}
```

Passagem de strings para subprogramas

Modo de passagem

- A passagem de uma string é igual à passagem de um array, pois uma string é um array de carateres

Exemplos

```
int  comp (char  s[]);
int  comp (char  *s);
```

Subprogramas de manipulação de strings

Principais subprogramas

- Biblioteca

  string.h

- Subprogramas/funções

int strcmp (const char ***str1**, const char ***str2**);

  - compara alfabeticamente as strings apontadas por **str1** e **str2**
  - devolve um valor inteiro menor do que 0 (se str1 < str2), 0 (se str1 = str2), ou um valor inteiro maior do que 0 (se str1 > str2)

char ***strcpy** (char ***destino**, const char ***origem**);

  - copia a string apontada por **origem** para a string apontada por **destino**
  - devolve um ponteiro para a string apontada por **destino**

int strlen (const char ***str**);

  - devolve o comprimento (numero de carateres) da string apontada por **str**

Principais subprogramas

- Subprogramas/funções

char ***strcat** (char ***destino**, const char ***origem**);

- concatena duas strings

  (acrescenta a string apontada por **origem** ao fim da string apontada por **destino**)

- devolve um ponteiro para a string apontada por **destino**

char ***strchr** (const char ***str**, char **ch**);

- procura a primeira ocorrência do caráter **ch** na string apontada por **str**
- devolve o endereço da primeira ocorrência do caráter **ch** na string apontada por **str**, ou NULL se o caráter não existe

char ***strrchr** (const char ***str**, char **ch**);

- procura a última ocorrência do caráter **ch** na string apontada por **str**
- devolve o endereço da última ocorrência do caráter **ch** na string apontada por **str**, ou NULL se o caráter não existe

A palavra reservada const na passagem de parâmetros

- Exemplo da função **strcpy** (copiar uma string para outra)

```
char *strcpy (char *destino, const char *origem);
```

  - a palavra reservada **const** impede que a string apontada por origem seja alterada dentro do subprograma strcpy