A linguagem C

- Interacção com o Utilizador
  - Escrita /Leitura Formatada: printf(), scanf()
  - Especificadores de Formato
  - Escrita /Leitura não formatada
    - putchar(), puts()
    - getchar(), gets()

Interacção com o Utilizador

A função `printf()`

- Permite escrever valores no monitor
- Encontra-se definida no ficheiro **stdio.h**, portanto é necessário colocar a linha

```
#include <stdio.h>
```

- **Sintaxe**

```
printf(string_formatação,expr_1,expr_2,...,expr_n);
```

A função `printf()`
A *string* de formatação

- é colocada entre aspas
- composta por um conjunto de caracteres + especificadores
- existe um especificador por cada expressão da lista

Exemplos (sem especificadores de formato):

```
printf("1- Engenharia Electrotécnica...");
printf("2- Engenharia Electrotécnica...\n");
printf("3- Engenharia \n Electrotécnica...\n");
```

A função `printf()`
O especificador de formato `%`

- Necessário se pretendermos escrever no monitor um determinado valor (constante, variável, ou expressão).
- O valor a escrever, resultado da expressão, tem de ser convertido num conjunto de caracteres antes de poder ser escrito no monitor.
- O formato pretendido é indicado no especificador de formato, sendo este composto por, pelo menos, um caracter precedido de `%`
- A cada expressão na lista corresponde obrigatoriamente um especificador de formato.

A função printf()
O especificador de formato %

Especificador de formato (forma genérica)

`% [flags][width][.prec][l|L]` **`type_char`**

type_char	Tipo
d	inteiro c/ sinal
i	inteiro c/ sinal
x	inteiro s/ sinal em hexadecimal (a .. f)
X	inteiro s/ sinal em hexadecimal (A .. F)
o	inteiro s/ sinal em octal
f	real no formato parte inteira, ponto e parte fraccionária
e	real no formato exponencial, com caracter **e**

A função printf()
O especificador de formato %

type_char	Tipo
E	real no formato exponencial, com caracter **E**
g,G	escolhe melhor maneira de exibição entre normal e exponencial
c	caracter
s	*string* (cadeia de caracteres)
%%	caracter % (caso especial)

A função `printf()`
O especificador de formato %

`width`	**reserva um determinado número de colunas**
`n`	reserva **n** colunas, o valor é escrito alinhado à direita
`0n`	reserva **n** colunas, o valor é escrito alinhado à direita e acrescenta zeros à esquerda

`.prec`	**nºcolunas para a parte fraccionária(float e double)**
`nada`	para `f`, `e`, `E` coloca 6 casas decimais; apenas os dígitos significativos em `g`, `G`
`.n`	**n** casas decimais
`.0`	não coloca casas decimais (nem o ponto)

A função `printf()`
O especificador de formato %

`flags`	
`-`	altera justificação para justificado à esquerda
`+`	valores positivos são precedidos pelo caracter +
`espaço`	valores positivos são precedidos por um espaço

`[h,l,L]`	
`h`	`short int ou unsigned short int`
`l`	`long int` ou `double`
`L`	`long double`

Exemplo #1 - A função `printf()`

```
...
int i = 5;
long int j = 1024;
char let, B;
let='B';
printf("i=%d\n", i);
i=5
printf("Um inteiro=%d e um inteiro longo =%d\n", i, j);
Um inteiro=5 e um inteiro longo =1024

printf("Caracter %c na posição %d\n", let, let);
Caracter B na posição 66

...
```

Exemplo #2 - A função `printf()`

```
...
float x = 10.0;
double y = 758.625;
printf("N° real,vários formatos %f %e %E\n", x, x, x);
N°   real,vários   formatos   10.000000   1.000000e+01
1.000000E+01
printf("Outro real %f \n", y);
Outro real 758.625000
printf("Real %g  %G\n", y, y);
Real 758.625  758.625
printf("Um inteiro e um real %d  %f\n", -32, fabs(-32.0/3));
Um inteiro e um real -32  10.666667
...
```

Exemplo #3 - A função `printf()`

```
int i = 5;
double y = 758.625;
printf("Número=%10d\n", i);
Número=---------5
printf("%10c%12.1f\n", 'y', y);
---------y-------758.6
printf("%12.0f%012.0f\n", y, y);
---------759000000000759
printf("%12.1e\n", y);
-----7.6e+02
```

Interacção com o Utilizador

A função `scanf()`

- Permite ler valores a partir do teclado
- Encontra-se definida no ficheiro **stdio.h**, pelo que é necessário colocar a linha

```
#include <stdio.h>
```

Sintaxe

```
scanf(string_de_formatação,lista_de_endereços_variáveis);
```

A função `scanf()`
A *string* de formatação

- é colocada entre aspas;
- composta apenas por especificadores de formato;
- O endereço de uma variável é indicado colocando o caracter `&` antes do identificador.

Exemplos:

```
int a;
char let;
float real;
scanf("%d",&a);
scanf("%c",&let);
scanf("%f",&real);
```

A função `scanf()`
O especificador de formato (1)

Especificador de formato (forma genérica)

```
% [*][width][h|l|L] type_char
```

type_char	Tipo
`d`	inteiro
`i`	inteiro
`x`	inteiro s/ sinal em hexadecimal (a .. f)
`o`	inteiro em octal
`f, e, g`	real em vírgula flutuante
`c`	caracter
`s`	Cadeia de caracteres (*string*)

A função `scanf()`
O especificador de formato (2)

width	
n	É o número máximo de caracteres a serem lidos relativamente a uma determinada variável da lista

[h,l,L]

h	short int ou unsigned short int
l	long int ou double
L	long double

Exemplo #1 - A função `scanf()`

```
int i;
float x;
double y;
char let;
printf("Introduza uma letra");
scanf("%c", &let);
printf("Introduza um inteiro");
scanf("%d", &i);
printf("Introduza 2 números reais");
scanf("%f%lf", &x, &y);
printf("Inteiro =%d\n", i);
printf("Reais = %f\t%lf\n", x, y);
printf("Letra %c \n", let);
```

Leitura não Formatada

- **Leitura**:
  - getchar() -> permite ler um carácter

    ```
    char letra;
    letra=getchar();
    ```

  - gets() -> permite ler um conjunto de caracteres (*string* *)

    ```
    char frase[20];
    gets(frase);
    ```

* Vamos estudar mais tarde

Escrita não Formatada

- **Escrita**:
  - putchar() -> permite escrever um carácter

    ```
    putchar('E');
    ```

  - puts() -> permite escrever um conjunto de caracteres *string* *

    ```
    puts("Escrever uma frase!");
    ```

* Vamos estudar mais tarde