
Módulo	0784 - Programação em C/C++ - Funções e Estruturas
Local	
Sessão	
Formador	
Ficha	1 - Funções

Funções

Um programa escrito em linguagem C faz sempre uso de funções, sejam funções da biblioteca padrão, como o caso do *printf*, do *scanf*, do *getchar*, etc., seja através do uso da função *main*. Sendo esta função obrigatória, podemos dizer que um programa em C tem sempre, pelo menos, uma função.

Mas as funções padrão nem sempre são suficientes e pode ser necessário criar as nossas próprias funções. Pode ser também importante dividir o nosso código em funções de modo a melhorar a organização do nosso código e assim facilitar a escrita e manutenção do programa.

Funções permitem agrupar instruções que possuem uma relação lógica entre si, por exemplo todas as instruções necessárias para ler os dados pessoais de um utilizador, de modo a criar pequenos sub-programas com objectivo bem definido e limitado e que podem depois ser usadas como peças na construção de um programa maior.

Se pensarmos na função *printf*, o objectivo da função é pequeno e específico: mostrar texto no ecrã. É também uma função que isolada não produz resultado útil e que é sempre usada para ajudar na criação e outras funções ou programas. Não sabemos também como é que a função *printf* está implementada, mas isso não é importante para que a possamos usar.

Escrita de Funções

Para escrevermos as nossas funções é necessário duas coisas: declarar a função e implementar a função. Algo semelhante ao uso de variáveis, quando pretendemos criar uma função somos obrigados a declarar a função, situação onde indicamos **o nome da função, o tipo de retorno e o(s) tipo(s) do(s) parâmetro(s) que a função recebe**. Depois da declaração é necessário implementar a função, isto é, **escrever o código que a função vai conter** e que lhe confere a funcionalidade.

Estas duas partes, declaração e implementação são feitas em locais separados, mas sempre no nosso ficheiro principal com o código. No topo do ficheiro, imediatamente **antes da função main**, colocamos as declarações. No fundo do ficheiro, imediatamente **depois da função main**, colocamos a implementação das nossas funções. Embora o local onde colocamos as declarações e implementações possa ser diferente do que foi apresentado, durante este módulo iremos seguir sempre esta regra.

Exemplo.

```
float dobro(float valor);

int main()
{
    float x, resultado;

    printf("Insira um numero: ");
```

```

scanf("%f", &x);

resultado = dobro(x);

printf("Dobre de %.3f: %.3f", x, resultado);

return 0;
}

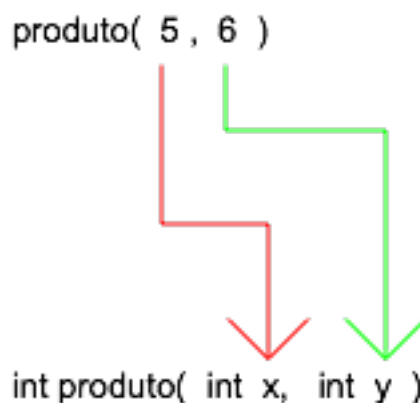
float dobro(float x)
{
    return 2 * x;
}

```

Parâmetros de Funções

A passagem de parâmetros para as nossas funções deve respeitar a declaração das mesmas, incluindo a ordem dos parâmetros. Devemos ter atenção ao facto de que ao passarmos parâmetros para dentro das nossas funções estamos a enviar cópias dos valores e que os valores originais nunca serão mudados ¹.

Figura 1. Relação entre Parâmetros e Variáveis Locais



1. Verdadeiro ou Falso

- Uma função pode devolver ao mesmo tempo mais que um valor;
- Uma função pode não ter parâmetros;
- Uma função tem que devolver sempre um inteiro;
- Os parâmetros das funções podem ser do tipo void;
- A instrução *return* termina uma função;
- Uma variável local a uma função pode ter o mesmo nome que um parâmetro;
- A instrução *return* termina uma função apenas e for a última instrução da função;

¹Mais tarde veremos que esta situação não é exactamente assim.

-
- h. A instrução *return* quando executada dentro de qualquer função termina o programa;
 - i. A instrução *return* quando executada dentro da função *main* termina o programa;
 - j. O nome de uma função é opcional;
 - k. Os parâmetros de uma função são opcionais;
 - l. O nome de uma função segue as mesmas regras que o nome de uma variável;
 - m. O número de linhas de uma função deve ser fixo e sempre inferior a 10
 - n. O nome de uma função pode ser igual ao de uma palavra reservada da linguagem C;

2. Programar

Implemente um único programa, composto por um menu, que permita executar todas as operações indicadas de seguida. Deve implementar cada uma das funcionalidades numa função diferente. Comece por implementar o menu e confirmar que o mesmo funciona adequadamente.

1. Ler cinco palavras;
2. Calcular o dobro de um valor;
3. Calcular o produto de dois valores;
4. Calcular a área de um rectângulo, $A = \textit{Altura} * \textit{Largura}$;
5. Calcular a área de um triângulo, $A = (\textit{Base} * \textit{Altura}) / 2$;
6. Calcular a área de uma circunferência, $A = 3.14 * \textit{Raio} * \textit{Raio}$;
7. Calcular a raiz de um número, use a função *sqrt()* disponível através da inclusão de *math.h*;
8. Simulador de Euromilhões, use a função *rand()* e *srand()*;
9. Conversão entre escudos e euros. 1 euro são 200.482 escudos;
10. Ler notas de um aluno e calcular a média;
11. Indicar o menor de dois valores;
12. Indicar o menor de três valores;
13. Calcular o factorial de um número. Factorial de 5 é $5 * 4 * 3 * 2 * 1 = 120$;
14. Emissão de cinco *beeps*;
15. Funcionalidade que verifique se um carácter é um dígito;
16. Funcionalidade que verifique se um carácter é uma consoante;
17. Funcionalidade que verifique se um carácter é uma vogal;

-
18. Funcionalidade que leia caracteres continuamente, termine de ler quando for inserido o #, e que indique quantos foram os dígitos, quantas as consoantes e quantas as vogais;
19. Funcionalidade que diga se um carácter é minúsculo, se for maior ou igual a a e menor ou igual a z então é minúsculo.
20. Funcionalidade que converta uma letra para maiúscula se a mesma for minúscula, converter para maiúscula = carácter + $A - a$;
21. Potência de ordem n de um número. A potência de ordem 5 de 10 é $10 * 10 * 10 * 10 * 10 = 100000$;