FACULDADE DE CIÊNCIAS E TECNOLOGIA
UNIVERSIDADE NOVA DE LISBOA

Departamento de Engenharia Electrotécnica

PROGRAMAÇÃO DE MICROPROCESSADORES

2021 / 2022

Mestrado Integrado em Engenharia Electrotécnica
e Computadores

1º ano
1º semestre

Trabalho nº 5
Funções

http://tele1.dee.fct.unl.pt/pm

Paulo da Fonseca Pinto
Luis Bernardo

1 Introdução

O capítulo 5 do livro "Linguagem C" de Luís Damas, recomendado para a disciplina de Programação de Microprocessadores, é dedicado às funções. Esta aula visa consolidar estas matérias através de um conjunto de exercícios. Faça todos os exercícios pedidos em ficheiros separados e **GUARDE O CÓDIGO desenvolvido na memória USB**. Durante a aula o docente pode pedir-lhe para mostrar o código desenvolvido.

2 Como escrever um ficheiro com funções?

2.1 Onde colocar as funções?

Uma pergunta pertinente quando se escreve o código de um programa com funções é saber onde colocar as funções e o ***main***. Deve ser como está mostrado em baixo à esquerda, ou à direita?

```
#include <stdio.h>

main ()
{
  int i= 4, n=2;

  escreve_num (i, n);
}



void escreve_num (int a, b)
{
  printf ("N1 = %d, N2 0 %d\n",
                    a, b);
}
```

```
#include <stdio.h>

void escreve_num (int a, b)
{
  printf ("N1 = %d, N2 0 %d\n",
                    a, b);
}



main ()
{
  int i= 4, n=2;

  escreve_num (i, n);
}
```

PORQUÊ?

O compilador de C vai percorrendo o ficheiro do princípio ao fim, compilando o programa. Se usarmos o modo à esquerda, quando o compilador encontra a função `escreve_num`, como ainda não a conhece, admite que ela devolve `int`.

É assim que o compilador de C funciona!

Quando chega à função propriamente dita repara que existe uma função que devolve `void`, mas que tem o mesmo nome de uma outra que ele já conhece, só que esta devolve `int`. Ora só pode ser uma função diferente, mas não se pode ter funções diferentes com o mesmo nome. O compilador detecta um erro!

Existe uma solução para estes casos:
Escrever um **protótipo** da função `escreve_num` **antes** do `main` para indicar ao compilador que `escreve_num` devolve `void`.

Ora quanto mais se escreve, mais se erra! Se a meio de programar decidirmos mudar alguma coisa na função **temos de ir** mudar o protótipo.

Assim, para simplificar, vai-se fazer a seguinte regra:

> É proibido usar protótipos em Programação de Microprocessadores.
> Só se poderá usar no caso de se utilizar mais do que um ficheiro para escrever o código.

ENTÃO O QUE SE DEVE FAZER?

Deve-se usar o modo de escrita apresentado à direita na figura da página anterior.

> O `main` é sempre a última função no ficheiro e sempre que se chama uma função, chama-se "para cima" (isto é, o código dela está escrito em cima). Deste modo não são precisos os protótipos.

2.2 AS VARIÁVEIS GLOBAIS

Para este assunto não vão ser dadas muitas explicações:

> É proibido usar variáveis globais em Programação de Microprocessadores.

3 A tabuada

Pretende-se utilizar o código seguinte, sem alterar qualquer linha já escrita, para fazer a tabuada. Acrescente o que for preciso num ficheiro ***tabuada.c***.

```
/*
 * Tabuada
 * Ficheiro: tabuada.c
 */
#include <stdio.h>
#include <stdlib.h> /* define exit() para sair do programa */

int mult (int a, b)
{
   return a*b;
}

main () {
    int n;

    printf ("Que tabuada quer (1 a 9)? : ");
    if (scanf (" %d", &n) != 1) {     /* Se não leu um elemento */
        printf ("Leitura do limite inválida\n");
        exit (1); /* Sai do programa */
    }
    if ((n < 1) || (n > 9)) {
        printf (" Número incorrecto. Tente outra vez\n");
        exit (1);
    }

    tabuada (n);
}
```

O programa deve proceder como este exemplo. Pode depois mudar o código para ficar sempre a perguntar.

```
Que tabuada quer (1 a 9)? : 5

  5 x 1 = 5
  5 x 2 = 10
  5 x 3 = 15
  5 x 4 = 20
  5 x 5 = 25
  5 x 6 = 30
  5 x 7 = 35
  5 x 8 = 40
  5 x 9 = 45

Que tabuada quer (1 a 9)? :
```

4 Exercício Final: "Jogo do mais ou menos"

Este exercício foi baseado num exercício semelhante de anos passados da autoria de Yves Rybarczyk e Vasco Gomes.

4.1 DESCRIÇÃO GERAL

O objectivo é construir um jogo para que o utilizador descubra um número mistério. O nível do jogo, o número máximo de tentativas e a escolha do número mistério podem ser decididos por outro jogador, ou o próprio programa pode calculá-los por sorteio.

Existe assim uma fase de **preparação** de jogo, e uma fase de **execução** de jogo. No final da execução, o programa deve perguntar se se deve repetir tudo.

Repare que a fase de execução do jogo **é igual para os dois casos** (haver um outro jogador a decidir as coisas, ou ser o próprio programa).

O programa deve começar por perguntar se a escolha deve ser feita por uma pessoa, ou automaticamente.

No caso de ser por uma pessoa o programa pergunta o máximo de tentativas, o nível do jogo e o número mistério dentro do intervalo do nível escolhido. No caso de ser automático deve usar-se a função `rand` para se gerar os números pseudo-aleatórios.

A função rand gera sempre a mesma sequência de números aleatórios dependente da semente que esteja definida. Isto é, para a mesma semente a sequência de números gerados é sempre a mesma. Este comportamento é útil quando se quer comparar algoritmos diferentes com os mesmos valores "aleatórios".

Para se gerar uma nova semente usa-se a função `srand`. Assim, os alunos devem chamar a função `srand` no `main` para obter uma nova semente (a partir dos últimos bits do relógio como está mostrado abaixo), e usar a `rand` para obter um valor entre `MIN` e `MAX` usando o que está mostrado em baixo. Não esquecer de incluir os ficheiros de declaração necessários. Para mais esclarecimentos consulte o livro (pág. 221).

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

srand (time(NULL));

numero_misterio = (rand () % (MAX - MIN + 1)) + MIN;
```

Depois da fase de preparação, o programa deve limpar o ecrã no caso da escolha ter sido manual, e escrever o nível do jogo (também com o intervalo de valores possível) e o número máximo de tentativas.

A seguir começa a fase de execução do jogo em que a cada tentativa do jogador o programa deve indicar se o número mistério está para cima ou para baixo do valor introduzido e também o número de tentativa que faltam.

4.2 VALORES POSSÍVEIS

Existem três níveis de jogo:

- Nível 1: número mistério entre 1 e 100
- Nível 2: número mistério entre 1 e 1000
- Nível 3: número mistério entre 1 e 10000

Existem três valores para o número máximo de tentativas:

- Tentativas 1: 10
- Tentativas 2: 15
- Tentativas 3: 20

4.3 OBRIGATORIEDADE

Os alunos têm obrigatoriamente que obedecer às seguinte indicações

1. A função `main` é muito pequena e quase que só faz chamadas a funções. Pode conter apenas alguma (ou algumas) instrução `if` fundamental.

2. Devem existir os menus suficientes para informar os utilizadores do que é preciso ele introduzir (fora da `main`). As funções desses menus podem ser elas mesmas a ler os valores introduzidos pelo utilizador.

3. Cada opção dos valores a introduzir (nível, tentativas, mistério) é executada numa função própria para essa opção.

4. Existe APENAS uma função que executa a escolha de um número mistério entre MIN e MAX. Essa função será utilizada para os vários propósitos (nível, tentativas, mistério).

5. Repare que malgrado ser sorteio ou escolha por um utilizador, o jogo é o mesmo, e deve haver apenas uma função chamada `jogo`.

Um exemplo para o programa no caso de sorteio está mostrado na página seguinte (para o caso manual as alterações são óbvias):

```
/*********  JOGO DO SOBE E DESCE  **************/

Valores por sorteio (1) ou manuais (2):  4
Erro na introdução de dados.

Valores por sorteio (1) ou manuais (2):  1

Estamos a jogar com o nível 2 (1 a 1000) e com um máximo de tentativas de 20.

Diga um número:  50
O numero mistério está abaixo.
Ainda tem 19 tentativas.

Diga um numero:  30
Parabéns!
Acertou em 2 tentativas

Quer jogar outra vez (s ou n): n
Adeus.
```