FACULDADE DE CIÊNCIAS E TECNOLOGIA
UNIVERSIDADE NOVA DE LISBOA

Departamento de Engenharia Electrotécnica

PROGRAMAÇÃO DE MICROPROCESSADORES

2021 / 2022

Licenciatura em Engenharia Electrotécnica
e Computadores

1º ano
1º semestre

Trabalho nº 3
Testes, condições e ciclos (Parte 1)

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

Paulo da Fonseca Pinto
Luis Bernardo

1 Introdução

Nos capítulos 3 e 4 do livro "Linguagem C" de Luís Damas, recomendado para a disciplina de Programação de Microprocessadores, são apresentados respectivamente os testes e condições, e os ciclos. Esta aula, e a próxima, visam 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 TESTES E CONDIÇÕES

Ao contrário de outras linguagens, a linguagem C não define um tipo específico para guardar valores Booleanos (Verdadeiro/Falso). É normalmente utilizado o tipo inteiro, onde Falso é representado pelo valor 0 (zero), e verdadeiro, por qualquer valor diferente de 0.

Os valores dos tipos básicos apresentados nos exercícios da última aula podem ser comparados utilizando-se as operações: > (maior), >= (maior ou igual), < (menor), <= (menor ou igual), == (igual), e != (diferente). Os valores Booleanos podem ser combinados utilizando as operações && (AND), || (OR), ou ! (Negação).

As condições Booleanas podem ser testadas, e pelo resultado mudar a sequência do programa executado. A instrução base é a sequência

"***if (cond) then-expr1; else expr2;***"

ou simplesmente

"***if (cond) then-expr1***;"

Caso se queiram utilizar várias instruções em *expr1*, e *expr2*, é necessário colocar estas expressões entre chavetas **{ }**. Existem duas formas compactas de teste, baseadas na instrução ***switch***, e no operador ternário ***?:***, com os significados representados na figura abaixo.

Deve-se sempre **indentar** o código dentro de chavetas, e dentro de uma instrução ***if***, ***else***, ou ***switch***, de forma a torná-lo mais legível.

Repare como foi feita a indentação no código da página seguinte.

```
switch (var) {
    case valor1:  expr1;
                  break;
    case valor2:  expr2;
                  break;
    default:      exprd;
}
```

é equivalente a

```
if (var == valor1) {
    expr1;
} else if (var == valor2)
{
    expr2;
} else {
    exprd;
}
```

```
var= cond ? expr1 : expr2;
```

é equivalente a

```
if (cond)
    var= expr1;
else
    var= expr2;
```

EXERCÍCIO 1: Crie um ficheiro novo com o nome ***data.c*** e introduza o código apresentado em baixo, omitindo os comentários (entre **/*** e ***/**). Este código lê duas datas como inteiros. Admita que não existem erros como dizer que Fevereiro tem 31 dias. O que se pretende é usar uma sequência de testes (***if*** e ***else***) para chegar à conclusão se a data 1 é anterior à data 2 ou não. Não é permitido passar tudo para dias e depois comparar.

Sistematize o uso de ***if*** e ***else*** em função da lógica das comparações. Aprenda a usar o ***else***: se perguntar a uma pessoa se é um homem e ela lhe disser que não, é capaz de ser desnecessário perguntar-lhe se é uma mulher…).
Complete o código com os testes necessários!

```
/*
 * Exercício 1 - cálculo da data anterior
 * Ficheiro: data.c
 */
#include <stdio.h>

main() {
    unsigned int ano1, mes1, dia1;
    unsigned int ano2, mes2, dia2;

    printf("Escreva para a data1 o dia mês ano como inteiros: ");
    scanf(" %d %d %d", &dia1, &mes1, &ano1);
    printf("Escreva para a data2 o dia mês ano como inteiros: ");
    scanf(" %d %d %d", &dia2, &mes2, &ano2);

    /* Calcula qual a data anterior */
   /* if (xxxxxxxxxxxxx) {

        /* EXERCÍCIO: COMPLETE O QUE FALTA AQUI! */

    }
}
```

EXERCÍCIO 2: Crie um ficheiro novo com o nome ***nota.c*** e introduza o código apresentado de seguida, omitindo os comentários (entre **/*** e ***/**). Este código pretende determinar a nota final à disciplina de Programação de Microprocessadores. No entanto, falta realizar duas regras:

a) Um aluno com duas faltas só pode ter no máximo 14 na nota de laboratório;

b) Um aluno com três faltas só pode ter no máximo 12 valores na nota final.

Complete o código com os testes necessários!

```
/*
 * Exercício 2 - cálculo da nota final de PM
 * Ficheiro: nota.c
 */
#include <stdio.h>

main() {
    /* Dados do aluno */
    unsigned int nota_teorica= 0, nota_pratica= 0, faltas= 0;
    float media;                /* média final */
    unsigned int nota_final;    /* nota final */

    printf("Cálculo da nota final a PM\n\n");
    printf("Introduza os seguintes dados:\n");
    printf("\tNúmero de faltas nos trabalhos práticos: ");
    scanf("%d", &faltas);
    printf("\tNota dos trabalhos práticos: ");
    scanf(" %d", &nota_pratica);
    printf("\tNota dos testes e exames: ");
    scanf(" %d", &nota_teorica);
```

```
    /* Calcula média */
    if (faltas > 3) {
        printf("Reprovou à disciplina por faltas\n");
    } else {
        /* EXERCÍCIO: CORRIJA O QUE FALTA AQUI!
                    Pode mudar este else todo*/
        media= 0.5*nota_pratica + 0.5*nota_teorica;
        /* Arredonda real a 0.5 e converte para int! */
        nota_final= (int)(media+0.5);
        printf("Aprovado com a nota final de %d valores (%.1f)\n",
               nota_final, media); /* Escreve só 1 casa decimal */
    }
}
```

3 Ciclos

Existem três instruções de **<u>controlo de fluxo</u>** em C:

while
for
do… while

Elas são muito parecidas e quase equivalentes. As diferenças acabam por ser bem subtis.

- A instrução *while* é usada para repetir a realização de outra instrução (ou grupo de instruções entre {}) <u>enquanto uma condição se mantiver verdadeira</u>.
- A instrução *for* é praticamente equivalente no que respeita à condição, <u>mas controla também o andamento de uma sequência</u>.
- Finalmente a *do…while* tem a grande característica de que a <u>condição é efectuada no final</u> fazendo com que a expressão do ciclo (ou expressões) seja efectuada pelo menos uma vez.

Dependendo do problema concreto a escolha de uma delas torna o programa mais "elegante". No entanto, como se disse, elas são muito equivalentes como se prova com os exemplos mostrados em baixo.

`for (init; cond; pos-inst)`
`    expr;`
	é equivalente a	`init;`
		`while (cond) {`
		`    expr;`
		`    pos-inst;`
		`}`

`do {`
`    expr;`
`} while(cond);`
	é equivalente a	`expr;`
		`while(cond) {`
		`    expr;`
		`}`

Os ciclos são muito usados em Programação e vamos dedicar duas aulas a eles (e às condições). Os ciclos servem para repetir certas acções quer de um modo exactamente igual, quer ao longo de um determinado conjunto (números naturais, elementos masculinos de uma população, todas as segundas-feiras de um calendário, etc., etc., etc.).

> Não se deve pensar que o objecto de utilização de ciclos tem sempre um carácter matemático. O aluno tem de começar a pensar em como "*dissecar*" o problema a programar utilizando as instruções que uma linguagem tem. Os ciclos e as condições são das instruções mais importantes para estruturar os problemas.

Por exemplo, veja o código em baixo e tente perceber o que ele pretende fazer. Será que funciona? Crie um ficheiro ***lerChar.c*** com este código, corra o programa e veja como funciona.

```
/*
 * Exercício 2 – Leitura de dados
 * Ficheiro: lerChar.c
 */
#include <stdio.h>
#include <stdlib.h>     /* define exit() para sair do programa */

main () {
    char c1;
    short int sucesso = 0;
    short int num;

    printf ("Escreva um 's':  ");
    while (!sucesso)  {
         scanf (" %c", &c1);
         if (c1 == 'f')
            exit (1);    /* sai do programa  */
         if (c1 == 's')
            sucesso = 1;
         else
           printf ("Vá lá, escreva um 's':  ");
    }
    printf ("Boa! Muito obrigado\n\n");
    printf ("Agora escreva um número inteiro: ");
    scanf (" %d", &num);
    printf ("O número foi %d\n", num);
}
```

EXERCÍCIO 3: Crie um ficheiro novo com o nome ***lerFloat.c*** vagamente baseado no código anterior que faça o seguinte:

Leia números reais muito rigidamente e sempre como caracteres. Os números reais são **SEMPRE** compostos por três algarismos da parte inteira, por uma vírgula, e por dois algarismos da parte fraccionária.

```
c c c , c c
```

Se o utilizador colocar algo diferente (por exemplo uma letra) o programa deve sair, mostrando uma mensagem indicando a razão porque acabou. Pode ser útil pensar numa instrução como a mostrada em baixo assumindo que se leu o carácter `k`.

```
if ((k>='0') && (k<='9'))
```

Se o utilizador colocar os seis caracteres correctos o programa deve apresentar o número na forma tradicional e na forma científica, usando uma variável ***float***, agradecer ao utilizador e sair.

MUITO IMPORTANTE

O programa deve ser feito com um ciclo. Em cada vez que o ciclo corre é lido ***um carácter*** (por exemplo o quarto carácter é uma vírgula). Consoante a "vez" do ciclo, esse carácter vale algo e vai-se construindo o ***float*** final.

No final escreve-se o valor.

4 Exercício final

4.1 CONTROLAR OS VALORES DE RETORNO DE *SYSTEM CALLS*

Nesta aula o exercício final é um pouco mais simples...

Mas para "compensar" vai-se dar **a partir de agora e até ao fim da disciplina de Programação de Microprocessadores** uma importância muito grande aos valores de retorno de funções de sistema.

A ideia é simples: Quando se chama uma função do sistema (como o ***scanf***) está-se a pedir para se fazer alguma coisa (ler valores). Ora, quando a função acaba a sua tarefa, como é que sabemos que o que pedimos foi feito? Isto é, que as variáveis que queríamos ler foram lidas?

Lembra-se no primeiro trabalho do *scanf* tentar ler um inteiro, ter sido digitado uma letra e tudo falhar?

Tomemos então o caso do ***scanf***. Se correrrmos a instrução **man** está lá escrito:

> "*devolve um inteiro a indicar o número de campos de entrada que foram lidos, convertidos e armazenados com sucesso. Este valor de retorno não inclui campos que foram lidos e não foram armazenados*".

Assim é de boa programação e deve ser usado por todos os alunos **daqui para a frente** o seguinte: sempre que se chama o ***scanf*** (ou outra função de biblioteca que retorna valores) deve-se guardar o valor de retorno numa variável e depois testar para ver se ela tem o valor que queremos.

Por exemplo, considere a instrução

```
result = scanf (" %d %d %d", &i1, &i2, &i3);
```

Se o utilizador escrever no teclado "**12 13 14**", o valor de `result` no final da instrução ter sido realizada vai ser de **3**. O que acha se a escrita do teclado for "**12 ola**"?

EXERCÍCIO 4: Use ficheiro ***mediaSIntFor.c*** do trabalho nº 2 que lê três valores de um modo muito rígido, e chame a este novo ficheiro ***lerMult.c***. Substitua o ***scanf*** pelo código mostrado em baixo que testa o valor de `result`. Experimente com várias entradas, válidas e não válidas.

```
result = scanf("%d %d %d", &val1, &val2, &val3);
printf ("Fora: Result = %d\n", result);
while (result != 3) {
  printf ("Erro na leitura dos dados.\n");
  printf ("Por favor escreva novamente os valores\n");
  result = scanf("%d %d %d", &val1, &val2, &val3);
  printf ("Dentro: Result = %d\n", result);
}

      continuar o programa pois os valores foram bem lidos
```

4.2 EXERCÍCIO FINAL

Observe o código incompleto em baixo que mostra um menu com várias hipóteses, e partes a amarelo onde não foi colocado mais nenhum código propositadamente. Pretende-se o seguinte: primeiro, o utilizador introduz o carácter de uma das escolhas. Depois é-lhe perguntado para repetir a escolha que fez antes desta, e ele tem de introduzir o carácter correspondente à escolha que fez antes desta. O programa faz então o seguinte:

- Ao ler a opção do utilizador para esta escolha, escreve o texto que se encontra na linha do menu a seguir à hipótese respectiva (*É pateta, nós sabemos*…). Por exemplo, se escreveu '*1*' deve aparecer o texto "*A opção que escolheu foi 1*".
- Pergunta ao utilizador que indique outra vez a hipótese que escolheu antes desta. Se o utilizador acertou escreve-lhe os parabéns. Se o utilizador falhou indica que ele falhou e escreve a opção que ele realmente escolheu antes desta.

Considera-se na primeira escolha, que a escolha anterior (que não existiu) tenha sido o carácter zero '0'.

MUITO IMPORTANTE: Para complicar um pouquito tem de fazer o seguinte: use o ***scanf*** para ler a opção presente e use o ***getchar*** para ler a opção anterior.

Crie um ficheiro novo com o nome ***menu.c***, comece com o código apresentado na figura abaixo completando-o para cumprir o que se pretende.

```
/*
 * Exercício Final - Menu
 * Ficheiro: menu.c
 */
#include <stdio.h>

main() {
    /* declaração de variáveis */


    /* escrever menu */
    do {
      printf ("              MENU para broncos\n\n");
      printf ("1 - A opção que escolheu foi 1\n");
      printf ("2 - Escolheu a segunda opção\n");
      printf ("3 - Agora escreveu um três\n");
      printf ("4 - A tecla que carregou foi um quatro\n");
      printf ("s - Escreveu um 's'\n");
      printf ("t - Esta foi a opção do 't'\n");
      printf ("f - Sair\n");

Pode ter código aqui

    } while(Pode ter código aqui

Pode ter código aqui

}
```

Um exemplo do programa correr é o seguinte (tenha em atenção as entradas a amarelo):

```
               MENU para broncos

1 - A opção que escolheu foi 1
2 - Escolheu a segunda opção
3 - Agora escreveu um três
4 - A tecla que carregou foi um quatro
s - Escreveu um 's'
t - Esta foi a opção do 't'
f - Sair

Escolha uma opção: 2

2 - Escolheu a segunda opção

Qual foi a opção que escolheu antes desta?  0
Muito bem

               MENU para broncos

1 - A opção que escolheu foi 1
2 - Escolheu a segunda opção
3 - Agora escreveu um três
4 - A tecla que carregou foi um quatro
s - Escreveu um 's'
t - Esta foi a opção do 't'
f - Sair

Escolha uma opção: t

t - Esta foi a opção do 't'

Qual foi a opção que escolheu antes desta?  2
Muito bem

               MENU para broncos

1 - A opção que escolheu foi 1
2 - Escolheu a segunda opção
3 - Agora escreveu um três
4 - A tecla que carregou foi um quatro
s - Escreveu um 's'
t - Esta foi a opção do 't'
f - Sair

Escolha uma opção: 3

3 - Agora escreveu um três

Qual foi a opção que escolheu antes desta?  3
Falhou! A opção era a t.
```