

INTRODUÇÃO À PROGRAMAÇÃO
RESOLUÇÃO DO EXAME FINAL DE SETEMBRO DE 1996
INF. DE GESTÃO/ENG^a E GESTÃO PRODUÇÃO/ENG^a E GESTÃO OBRAS

3ª Edição - 25.Junho.2008

Resolução e breves notas por Marco A.G.Pinto, Licenciado em Informática de Gestão.

© 1996-2008. Este documento é Freeware e pode ser livremente distribuído intactamente.

Breves notas:

Resolvi escrever este documento por não existir nada semelhante até à altura, nem de uma forma tão aprofundada, para que os alunos pudessem consultar, e daí fazer falta.

Esforcei-me não só por apresentar a resolução do exame de Setembro, como também me preocupei ao máximo em apresentar a resolução de uma forma o mais clara e simples possível e adicionar, sempre que possível, algumas informações úteis que permitam entender melhor como os programas funcionam, bem como aumentar os conhecimentos em relação à disciplina.

Para evitar repetir certos conceitos que aumentariam o tamanho deste documento e o tornariam mais papudo e maçador, existem coisas que só são explicadas de uma forma mais aprofundada uma vez pois o funcionamento é semelhante nos outros exercícios (leiam tudo!).

Na apresentação dos programas em PASCAL entende-se como uma boa apresentação:

- 1) Existem palavras que deverão ser escritas em maiúsculas e outras em minúsculas:
palavras reservadas = deverão ser escritas em maiúsculas, incluindo também o tipo das variáveis (INTEGER, REAL... etc.). Quando estiverem a utilizar o editor de PASCAL notarão que as palavras reservadas mudam de cor. Nesta resolução elas estão a “**Negrito**”(Bold).
variáveis = deverão ser escritas em minúsculas e não podem conter acentos nem espaços no nome. Caso queiram usar uma expressão que contenha mais de uma palavra e evidenciar isso, podem separar as palavras com “_”. Ex: my_cd
As variáveis deverão sempre começar por uma letra ou “_”.
outros comandos/procedimentos padrão = deverão ser escritos tal como as variáveis.
- 2) Os programas deverão ser sempre iniciados com a palavra “**PROGRAM**” seguida do título do programa em minúsculas e terminado com um “;”. O título do programa deverá ser escrito tal como explicado em relação às variáveis.
- 3) A seguir, se forem utilizadas UNITS, dever-se-á dizer quais. Ex:
PROGRAM marco;
USES crt;
As UNITS fornecem alguns comandos extra. A UNIT crt, citada no exemplo acima, permite pelo menos duas novas instruções úteis:
clrscr; - Limpa a tela com a cor do paper, default=preto (cor do EDITOR do DOS).
REPEAT
UNTIL *keypressed*; - Pausa o programa até que uma tecla seja premida.
- 4) A seguir deverão ser declaradas todas as variáveis utilizadas no programa bem como o respectivo tipo. Vejam acima como as apresentar e não esqueçam de inicializar as mesmas durante o programa antes de incidirem operações sobre elas.
- 5) O bloco (ou módulo) principal do programa é iniciado com a palavra “**BEGIN**” e termina com a palavra “**END.**”. Reparem que existe um ponto final a seguir a este **END.**

- 6) Durante a elaboração do programa devem Indentar o programa pois isso torná-lo-á mais agradável à vista e facilitará a sua compreensão. Indentar significa que as linhas com comandos não devem ser todas posicionadas na mesma coluna.
- 7) A validação dos dados é importante para assegurar que o programa não produza resultados incorrectos e realize a função para que foi feito.

Passando agora à resolução do exame propriamente dita:

(QUESTÃO – 1B)

```
PROGRAM somatorio;
VAR x,y,i : INTEGER;
      calc : REAL;
```

```
BEGIN
  write('Qual o valor de x? ');
  readln(x);
  REPEAT
    write('Qual o valor de y (y>=x)? ');
    readln(y);
  UNTIL y>=x;
  calc:=0;
  FOR i:=x TO y DO calc:=calc+2*i/10;
  writeln('CALC=',calc:0:2);
END.
```

Breves notas:

O que nos é pedido não é mais do que um programa que peça o valor do extremo inferior e superior do somatório, atendendo ao facto de o extremo superior dever ser igual ou superior ao inferior, e depois escrever uma rotina que nos permita calcular o valor do somatório dando de seguida, como resultado de saída, “CALC=<Valor do somatório>”.

O modo como um somatório funciona é simples:

Ex: $x=1$ $y=3$

$$CALC = \sum_{i=1}^3 2*i/10 = 2*1/10 + 2*2/10 + 2*3/10 = 0.2 + 0.4 + 0.6 = 1.2$$

Após o título do programa, começamos por declarar as variáveis que nele vão ser utilizadas:

x – Vai guardar o valor do extremo inferior do somatório.

y – Vai guardar o valor do extremo superior do somatório.

i – Vai ser utilizada para criar um ciclo entre x e y .

$calc$ – Vai ser utilizada para armazenar o resultado do somatório.

As variáveis x , y , i são do tipo inteiro ($-32768 \leq \text{INTEGER} \leq 32767$) pois esse dado é nos fornecido pelo enunciado (se x e y são do tipo inteiro, i também tem de o ser, ainda mais que o ciclo **FOR** só aceita números inteiros).

A variável $calc$ é do tipo real ($-2.9e-39 \leq \text{REAL} \leq 1.7e38$) porque dentro do somatório existe uma divisão do tipo “/”, que produz um resultado do tipo real (um número que pode ter vírgulas) necessitando de uma variável desse tipo para armazenar o resultado.

Após a declaração das variáveis, passa-se às rotinas dentro do módulo principal do programa:

Começa-se primeiro por ler a variável x (extremo inferior do somatório), pois ela pode assumir qualquer valor inteiro. É o valor y que está limitado e dependente do valor dado a x pois y tem de ser $\geq x$.

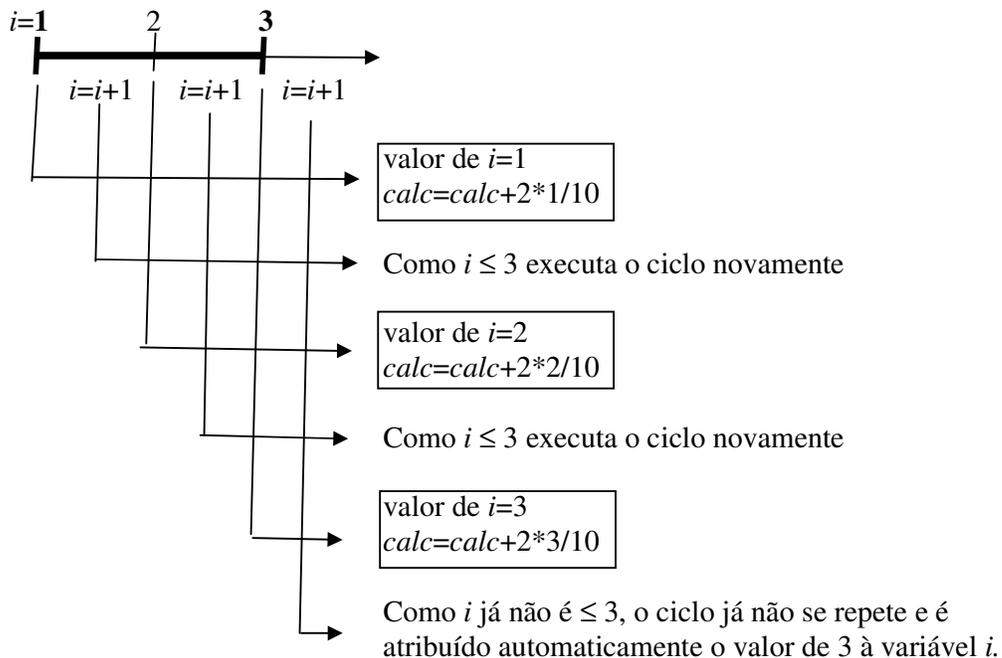
A seguir, lê-se o extremo superior (y) dentro de um ciclo **REPEAT/UNTIL** que tem como objectivo validar o valor introduzido pelo utilizador que tem de ser \geq ao de x . Enquanto esta condição não se verificar o computador irá repetir (**REPEAT**) a leitura de y até (**UNTIL**) $y \geq x$. Depois de lidos os extremos do intervalo já se pode passar ao cálculo do somatório cujo resultado vai ser depositado na variável $calc$:

Inicializa-se a variável $calc$ com o valor zero (atribui-se-lhe esse valor) pois não se pode fazer cálculos com variáveis que não tenham valor, se bem que nas versões mais recentes do TURBO PASCAL as variáveis declaradas tenham como valor default o zero.

Depois, para calcular o valor de $calc$, cria-se um ciclo **FOR**. O ciclo **FOR** diz-nos que uma variável inteira dada por nós (neste caso o i), está compreendida entre dois valores inteiros, um valor inicial e um valor final, e o(s) comando(s) dentro do ciclo **FOR** serão executados $\text{vf-vi}+1$ vezes e cada vez que são executados a variável i , que toma como valor inicial x , é incrementada uma unidade. Enquanto o valor de i for \leq ao valor final, o ciclo é repetido.

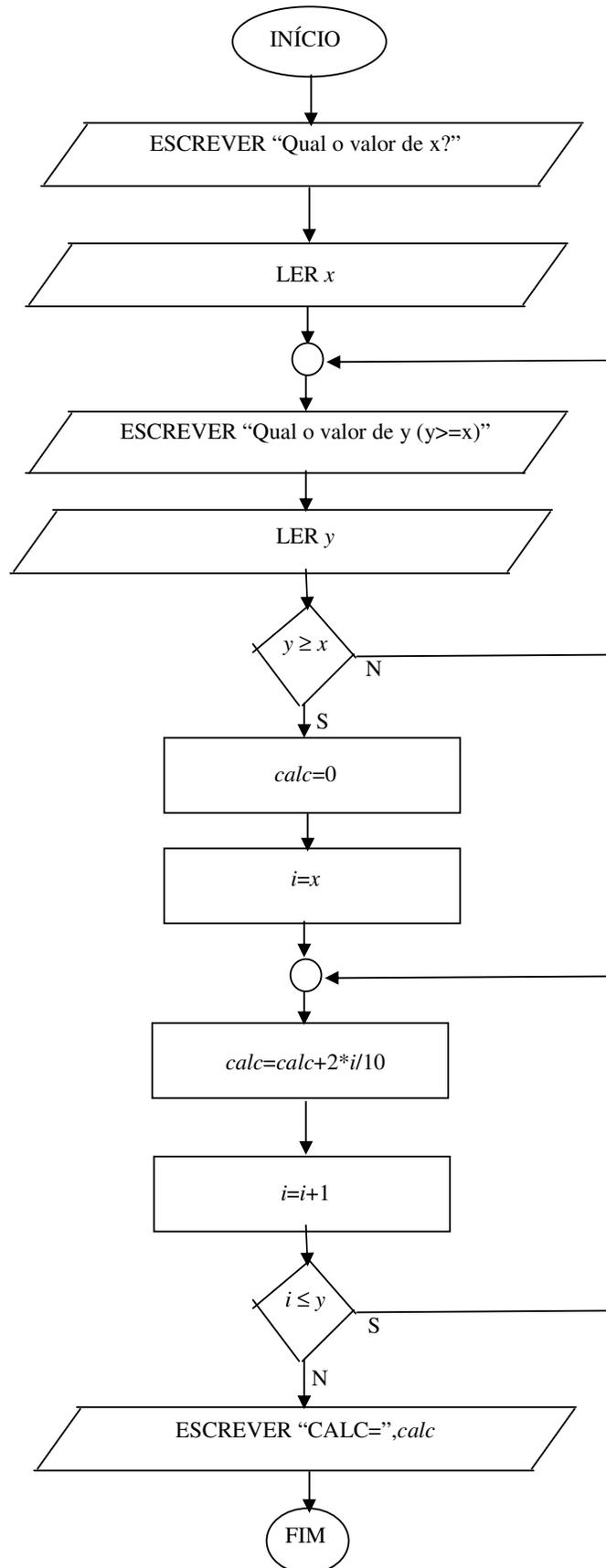
Após finalizado o ciclo, o valor da variável utilizada no ciclo **FOR** não fica com o $\text{v. final} + 1$ mas apenas com o valor final (o que não acontece noutras linguagens que utilizei).

Nesse caso, o ciclo **FOR** está compreendido entre x e y (os valores do extremo inferior e superior respectivamente). Como este ciclo vai funcionar e ainda com base no exemplo de cima: $x=1$ $y=3$, logo: $x \leq i \leq y$, ou seja: $1 \leq i \leq 3$, ou seja:



Depois, é só escrever no écran "CALC=" e mostrar o valor da variável $calc$. A expressão ":0:2", após a variável de saída de resultados, tem como objectivo tornar o resultado mais simpático e compreensível ao utilizador. Em vez de dar um resultado do tipo, por exemplo, "2.5300000000E+02" dá-nos como resultado um número real com apenas dois dígitos após a vírgula flutuante. Exemplo: 13.00 ou 144.23 ou 1.00 ou... etc.

(QUESTÃO – 1A)



Breves notas:

O fluxograma não é mais do que a representação gráfica simplificada de um programa através de símbolos, onde é evidenciado o fluxo de dados e as operações efectuadas sobre eles.

É aconselhável fazer-se primeiro o programa e só depois o fluxograma, pois o fluxograma não é mais do que uma cópia simplificada do programa e, além do mais, o grau de complexidade e tamanho do programa não são tão elevados que justifiquem que o programa só conseguisse ser bem concebido através de um fluxograma.

Penso não ser necessário grandes explicações relativamente ao que cada símbolo representa pois eles vêm nos apontamentos dados nas aulas.

(QUESTÃO – 2A)

```
PROGRAM media_aritmetica;
```

```
VAR x,y,f,soma : LONGINT;
```

```
BEGIN
```

```
  REPEAT
```

```
    write('Qual o valor de x (positivo)? ');
```

```
    readln(x);
```

```
  UNTIL x>0;
```

```
  REPEAT
```

```
    write('Qual o valor de y (y>=x)? ');
```

```
    readln(y);
```

```
  UNTIL y>=x;
```

```
  soma:=0;
```

```
  FOR f:=x TO y DO soma:=soma+f;
```

```
  writeln('A média aritmética é de ',soma/(y-x+1):0:2);
```

```
END.
```

Breves notas

O objectivo deste programa é pedir o início e o fim (os extremos) de um intervalo de números inteiros positivos e, de seguida, calcular a média aritmética dos seus elementos, ou seja, a soma de todos os elementos constantes no intervalo e, no final, a divisão do resultado da soma pelo número de elementos existentes no intervalo.

Após o título do programa, começamos por declarar as variáveis que nele vão ser utilizadas:

x – Vai guardar o valor do extremo inferior do intervalo.

y – Vai guardar o valor do extremo superior do intervalo.

f – Vai ser utilizada para criar um ciclo entre *x* e *y*.

soma – Vai ser utilizada para armazenar o resultado da soma de todos os elementos do intervalo.

As variáveis *x*, *y*, *f* são do tipo inteiro grande ($-2147483648 \leq \text{LONGINT} \leq 2147483647$) pois, embora o enunciado nos diga que sejam apenas do tipo inteiro, pensei na possibilidade de o intervalo também poder estar situado em valores mais altos que o INTEGER permita.

No final verão porque não perdi tempo a declarar uma variável do tipo REAL para armazenar o valor da média.

Após a declaração das variáveis, passa-se às rotinas dentro do módulo principal do programa:

Tal como no programa anterior, começa-se por ler a variável *x* primeiro pois *y* vai estar dependente do valor que for dado a *x*. Ao assegurar-se que *x* é um número positivo a única coisa que importa verificar em *y* é que este $e \geq x$, o que implicará que *y* também é um número positivo.

Para criar um valor x válido cria-se um ciclo **REPEAT/UNTIL**, dentro do qual está a leitura da variável x e este ciclo só será interrompido quando se verificar a condição que x é maior que 0 e por isso um número positivo.

Após obtido o valor x , pode-se agora ler o valor y tendo como base o facto de este ter de ser \geq ao de x . Para tal cria-se um ciclo **REPEAT/UNTIL**, dentro do qual se encontra a leitura de y , e este ciclo só é interrompido quando a condição de y ser maior ou igual a x se verificar.

Depois, passa-se ao cálculo da soma de todos os elementos pertencentes ao intervalo inicializando para tal a variável *soma* a zero. Depois crio um ciclo **FOR**, usando a variável f , compreendido entre x e y que correspondem ao primeiro elemento do intervalo e ao último respectivamente. Dentro do ciclo ponho uma instrução que vai adicionando ao valor da variável *soma* o valor de f ($x \leq f \leq y$). Quando o ciclo **FOR** terminar, a variável *soma* conterà a soma de todos os elementos do intervalo.

Depois, é só escrever o resultado da média no écran obtida por $\frac{soma}{(y-x+1)}$. O $(y-x+1)$ dá-nos o número de elementos existentes no intervalo e isto é fácil de entender bastando apenas materializar um intervalo com 1 elemento, por exemplo, $x=1$ e $y=1$. Neste intervalo apenas existe um elemento: {1}. Se para contarmos o número de elementos apenas fizéssemos $\frac{y-x}{y-x}$ isso daria zero, que é incorrecto. Por isso adicionamos +1. Se imaginarmos um intervalo $x=1$ $y=2$... etc. veremos que o mesmo se sucede.

Em vez de gastar uma variável do tipo REAL para armazenar o valor da média, ponho logo o resultado no écran pois este valor já não vai ter outro uso posterior. Para tornar o resultado mais simpático e compreensível ao utilizador formato a saída com “:0:2”.

(QUESTÃO – 2B)

```
PROGRAM multiplos7;  
VAR x,y,f,counter : LONGINT;
```

```
BEGIN
```

```
  REPEAT
```

```
    write('Qual o valor de x (positivo)? ');  
    readln(x);
```

```
  UNTIL x>0;
```

```
  REPEAT
```

```
    write('Qual o valor de y(y>=x)? ');  
    readln(y);
```

```
  UNTIL y>=x;
```

```
  counter:=0;
```

```
  FOR f:=x TO y DO IF f MOD 7=0 THEN counter:=counter+1;
```

```
  writeln('Número de múltiplos de 7: ',counter);
```

```
END.
```

Breves notas:

O objectivo deste programa é pedir o início e o fim (os extremos) de um intervalo de números inteiros positivos e de seguida contar o número de múltiplos de 7 existentes no intervalo e mostrar esse resultado.

Após o título do programa, começamos por declarar as variáveis que nele vão ser utilizadas:

x – Vai guardar o valor do extremo inferior do intervalo.

y – Vai guardar o valor do extremo superior do intervalo.

f – Vai ser utilizada para criar um ciclo entre x e y .

$counter$ – Vai ser utilizada para armazenar o número de múltiplos de 7 encontrados no intervalo. Sempre que um múltiplo de 7 é encontrado, esta variável é incrementada uma unidade.

As variáveis x , y , f são do tipo inteiro grande (LONGINT) pois, embora o enunciado nos diga que sejam apenas do tipo inteiro, pensei na possibilidade de abranger um intervalo maior àquele permitido pelo INTEGER.

O valor das variáveis x e y é obtido e validado da mesma forma que no programa anterior e por isso não é necessário explicar novamente como se deve proceder.

Depois, passa-se ao cálculo do número de múltiplos de 7 pertencentes ao intervalo inicializando para tal a variável $counter$ a zero (múltiplos de 7=0). Depois, crio um ciclo **FOR**, usando a variável f , compreendido entre x e y que correspondem ao primeiro elemento do intervalo e ao último respectivamente ($x \leq f \leq y$). Dentro do ciclo ponho uma instrução que verifica se o valor presente da variável f é múltiplo de 7 e se tal suceder incrementa em uma unidade a variável $counter$, que tal como o nome indica é utilizada como um contador.

Um número só é múltiplo de 7 se for divisível por 7 e, para verificar isso, divide-se o número por 7 ($N^\circ \text{ MOD } 7$) e se:

Resto da divisão = 0 \Rightarrow N° é múltiplo de 7

Resto da divisão \neq 0 \Rightarrow N° não é múltiplo de 7

A operação **MOD** só pode ser utilizada com os números inteiros (dá-nos o resto da divisão de dois números inteiros).

Findo o ciclo **FOR** só nos resta escrever no écran o número de múltiplos de 7 que se encontra na variável $counter$.

(QUESTÃO – 2C)

```
PROGRAM produto_acumulado;
VAR x,y,f,counter : LONGINT;
    produto : REAL;
BEGIN
  REPEAT
    write('Qual o valor de x (positivo)? ');
    readln(x);
  UNTIL x>0;
  REPEAT
    write('Qual o valor de y (y>=x)? ');
    readln(y);
  UNTIL y>=x;
  counter:=0;
  produto:=1;
  FOR f:=x TO y DO IF odd(f) THEN
    BEGIN
      produto:=produto*f;
      counter:=counter+1;
    END;
  IF counter=0 THEN writeln('Não existem números ímpares. ');
  IF counter=1 THEN writeln('Só existe um número ímpar que é o ',produto:0:0);
  IF counter>1 THEN writeln('O produto acumulado é: ',produto:0:0);
END.
```

Breves notas:

O objectivo deste programa é pedir o início e o fim (os extremos) de um intervalo de números inteiros positivos e, de seguida, calcular a multiplicação de todos os números ímpares nele existentes.

Neste problema colocam-se duas questões importantes:

- 1º E se não existirem números ímpares?
- 2º E se só existir um número ímpar?

Após o título do programa, começamos por declarar as variáveis que nele vão ser utilizadas:

- x* – Vai guardar o valor do extremo inferior do intervalo.
- y* – Vai guardar o valor do extremo superior do intervalo.
- f* – Vai ser utilizada para criar um ciclo entre *x* e *y*.
- produto* – Vai ser utilizada para armazenar a multiplicação dos números ímpares encontrados no intervalo. Sempre que um número ímpar é encontrado, o seu valor é multiplicado com o desta variável.

As variáveis *x*, *y*, *f* são do tipo inteiro grande (LONGINT) pois, embora o enunciado nos diga que sejam apenas do tipo inteiro, pensei na possibilidade de abranger um intervalo maior àquele permitido pelo INTEGER.

A variável *produto* é do tipo real pois a multiplicação de uma sequência de números cada vez maiores vai produzir um resultado enorme e em muitos casos um inteiro grande não conseguirá guardar tamanho valor.

O valor das variáveis *x* e *y* é obtido e validado da mesma forma que o programa anterior e por isso não é necessário explicar novamente como se deve proceder.

Depois, passa-se ao cálculo do produto acumulado e total de números ímpares existentes no intervalo inicializando para tal a variável *counter* a 0 e a variável *produto* a 1. Depois crio um ciclo **FOR**, usando a variável *f*, compreendido entre *x* e *y* que correspondem ao primeiro elemento do intervalo e ao último respectivamente ($x \leq f \leq y$). Dentro do ciclo ponho uma instrução que verifica se o valor actual da variável *f* é um número ímpar e, se tal suceder, incrementa em uma unidade a variável *counter*, que tal como o nome indica, é utilizada como um contador, e o número ímpar é multiplicado ao valor da variável *produto* (agora entendem porquê foi inicializada a um).

Notem que o raciocínio e técnica utilizados neste cálculo são os mesmos usados para calcular factoriais, a diferença maior reside no facto de só se fazerem multiplicações com números ímpares e de se ter uma variável (*counter*) para registar o número de multiplicações feitas.

Para ver se um número inteiro é ímpar, utilizou-se a função *odd(f)* que retorna o valor True se o valor de *f* for um número ímpar.

Findo o ciclo **FOR** vai-se passar à apresentação do resultado que passa por uma análise do conteúdo da variável *counter*:

- a) Se **counter=0**, significa que nenhum número ímpar foi encontrado e, por isso, não existe produto. É só escrever isto no écran (“Não existe produto.”).
- b) Se **counter=1**, significa que só um número ímpar foi encontrado e o valor deste encontra-se na variável *produto*. É só escrever no écran “Só existe um número ímpar que é:” e escrever o conteúdo da variável *produto* seguido de “:0:0” para formatar a saída de forma a que o número que saia não tenha vírgulas flutuantes (seja dado a entender ao utilizador que é um número inteiro).
- c) Se **counter>1**, significa que dois ou mais números ímpares foram encontrados no intervalo e só resta escrever no écran “O produto acumulado é:” e escrever o conteúdo da variável *produto* formatando a saída também com “:0:0” para evitar vírgulas flutuantes pois a multiplicação de números inteiros nunca dá números com vírgulas.

NOTA: A razão porque depois do **DO** de alguns ciclos **FOR** é utilizado um **BEGIN** e **END** é porque se está a utilizar mais de um comando dentro do ciclo **FOR** em questão.

(QUESTÃO – 3)

```
PROGRAM calcular_juros;
VAR n,f : INTEGER;
    si,i,sa : REAL;
CONST a_cobrar=250;
BEGIN
  REPEAT
    write('Qual o saldo inicial? ');
    readln(si);
  UNTIL si >0;
  REPEAT
    write('Qual o nº de meses? ');
    readln(n);
  UNTIL n>0;
  REPEAT
    write('Qual a taxa de juro anual(%)? ');
    readln(i);
  UNTIL i>0;
  i:=i/100/12;
  sa:=si;
  IF si<5000 THEN FOR f:=1 TO n DO sa:=sa-a_cobrar
                ELSE FOR f:=1 TO n DO sa:=sa+sa*i;
  IF sa<0 THEN writeln('A conta foi cancelada por atingir saldo negativo.')
            ELSE writeln('O saldo actualizado é de ',sa:0:2);
  IF sa<=si THEN writeln('Não houve juros.')
                ELSE writeln('Os juros foram de ',sa-si:0:2);
END.
```

Breves notas:

O objectivo deste programa é, tendo um capital inicial, uma taxa de juro mensal e um período de tempo durante o qual o capital vai ficar no banco, calcular quanto será esse capital ao fim do período de tempo bem como os respectivos juros.

O enunciado fala-nos ainda de uma sanção mensal de 250\$00 em caso do capital inicial ser inferior a 5000\$00.

Temos pois de obter o valor do capital inicial, o número de meses, a taxa de juro anual que depois temos de converter para mensal e calcular o valor actual desse capital findo o período de tempo estipulado e, em caso da conta ter atingido verbas negativas, indicar que a conta foi cancelada.

Após o título do programa, começamos por declarar as variáveis que nele vão ser utilizadas:

n – Vai guardar o valor do número de meses.

i – Vai guardar o valor da taxa de juro.

f – Vai ser utilizada para criar um ciclo entre 1 e *n*.

si – Vai guardar o valor do saldo inicial.

sa – Vai ser utilizada para armazenar o valor do saldo actual.

a_cobrar – corresponde à quantia descontada por mês (250\$00) caso o capital no banco seja inferior a 5000\$00.

As variáveis *n*, *f* são do tipo INTEGER pois *n* corresponde ao número de meses que é um número inteiro e *f* vai ser utilizado para criar um ciclo **FOR** entre o 1º e o último mês (*n*).

As variáveis si , sa , i são do tipo REAL pois si e sa são utilizadas para lidar com o capital no banco que pode ser um valor com centavos (ex: 10000\$67) e o i porque utilizam-se números com vírgulas nas taxas de juros.

A variável a_cobrar é uma constante pois tem um valor fixo que não vai ser introduzido nem alterado durante a execução do programa.

Começamos o programa com 3 ciclos **REPEAT/UNTIL** que vão ser utilizados para ler e validar variáveis: o 1º para o capital inicial (interrompido quando $si > 0$ pois tem de existir dinheiro no banco), o 2º para o número de meses que o capital vai ficar no banco (interrompido quando $n > 0$), o 3º para a taxa de juro anual (interrompido quando $i > 0$ pois as taxas de juros no banco nunca podem ser nulas ou negativas).

Após isto, como não se podem fazer contas usando o símbolo “%”, foi necessário converter a taxa de juro para um valor. Por exemplo: 15% são 0.15 obtidos ao dividir 15 por 100. Depois divide-se por 12 (pois o ano tem 12 meses) para converter a taxa para mensal.

No programa, $i := i / 100 / 12$ passa uma taxa anual em percentagem para uma taxa mensal já com um valor real e utiliza a própria variável i para a armazenar.

A seguir ponho $sa = si$ pois o valor de si vai ser necessário no fim do programa para o cálculo do juro e por isso não pode ser alterado. É sobre a variável sa que vão incidir as operações de variação de capital durante a execução do programa.

Começo por ver se vou aplicar a sanção de 250\$00 ou se vou adicionar juros mensais. Isto é verificado através de uma condição **IF** para ver se o capital inicial é inferior a 5000\$00. Se o resultado for:

TRUE = significa que o capital inicial é inferior a 5000\$00 e crio um ciclo **FOR**, compreendido entre 1 (1º mês) e n (último mês) e sempre que o ciclo é executado, o capital que está no banco (sa) é diminuído em 250\$00.

SENÃO (**ELSE**) (;se a condição $si < 5000$ não se verificar, for falsa)
= também crio um ciclo **FOR** mas neste caso cada vez que o ciclo é executado é adicionado juros ao capital que tínhamos no banco (sa - vai sendo aumentado cada vez que o ciclo é executado).

Após ter-se adicionado ou retirado dinheiro ao saldo actual todos os meses, resta agora saber que tipo de resposta se vai dar ao utilizador em relação ao saldo no banco:

Verifica-se então se o saldo actual é negativo ou não: Se for negativo ($sa < 0$) diz-se que a conta foi cancelada, senão (**ELSE**) diz-se qual o saldo actual pondo o valor da variável sa no écran e formatando a saída.

Agora vai-se informar o utilizador em relação aos juros, fazendo para isso uma comparação do saldo actual com o saldo inicial. Se o saldo actual for \leq ao saldo inicial isso significa que não houve juros e escreve-se isso no écran, senão (**ELSE**) diz-se qual o valor dos juros que é igual ao saldo actual menos o saldo inicial e formata-se a saída.