TÓPICOS DAS AULAS TEÓRICAS
de
Algoritmia e Programação

Algoritmia

por Maria da Conceição Neves

Desenvolvimento de Software

- O desenvolvimento de software é uma actividade de crescente importância na sociedade actual – Sociedade da Informação e do Conhecimento.
- A utilização de computadores nas mais diversas áreas de conhecimento humano requer o desenvolvimento de soluções computadorizadas cada vez mais sofisticadas e complexas.
- À medida que a complexidade dos problemas aumenta há necessidade de utilizar uma abordagem baseada nos princípios de engenharia.

A Engenharia de Software

- Na década de 70 há o reconhecimento de um conjunto de situações que são identificadas como "Crise de Software". Entre elas estão o crescimento dos custos de software e a sua fraca fiabilidade.
- Era preciso transformar a tarefa de construir software numa actividade com rigor comparável ao utilizado nas diversas áreas de engenharia. Surge assim uma nova disciplina a "Engenharia de Software".
- A engenharia de software tem como objectivo a produção de software de modo eficiente em termos de custo e fiabilidade. A engenharia de software é a aplicação prática do conhecimento científico nas fases de **especificação, análise, concepção (design), programação/implementação e manutenção** de software.

Maria da Conceição Neves

O objectivo de APROG

- Um dos objectivos da disciplina de APROG é iniciar os estudantes na produção de software.
- Como primeira disciplina nesta área de conhecimento começa-se por **desenvolver competências de raciocínio lógico através da programação e estruturação de dados.**
- Mas produzir software não é só programar …

Maria da Conceição Neves

Produzir software não é só Programar

- Do **Levantamento e Especificação de Requisitos** à criação de um **Produto de Software em funcionamento** há muito mais a fazer do que **Programação**
- **As fases de Análise e Concepção são muito relevantes**
- No entanto, na **resolução de problemas simples**, cujos **requisitos estão especificados**, geralmente num texto, **parte-se de imediato para a concepção de um algoritmo e estrutura de dados e respectiva codificação numa linguagem de programação**.

  O programa deve ser validado utilizando um adequado plano de testes.

Fases de Concepção e Programação

Concentremo-nos nas fases de Concepção e Programação baseando-nos na metodologia da programação estruturada.

Faz-se a decomposição do problema em problemas mais simples, que consiste em identificar os módulos (as funcionalidades) a serem executadas e definir a estrutura de controlo entre eles.

O método de decomposição deve ser um método descendente (top down) com as seguintes fases:

Decompor o **problema** em **módulos** integrando-os numa estrutura de controlo

(Descrever o respectivo algoritmo de integração dos módulos)

Para cada módulo

Se (o módulo é simples)

então

Programar o Módulo

(Descrever o respectivo algoritmo)

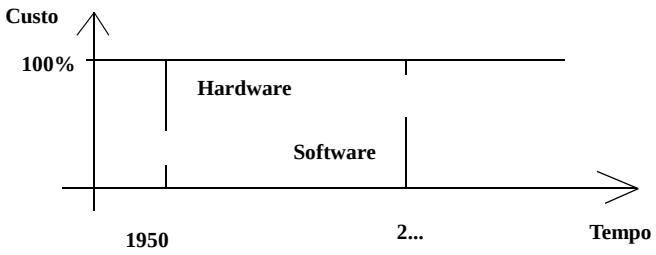
senão

Continuar a decompor o módulo segundo o mesmo princípio

Codificar segundo metodologia da programação estruturada **e Testar**

Maria da Conceição Neves

Desenvolvimento de Software



- Concepção e Programação Procedimental e Estruturados
  - Concepção Modular
  - Refinamento gradual (top-dwn) do topo para a base
  - Programação Estruturada

Programação Procedimental

Programação orientada à Acção

PROGRAMA =

Algoritmo

+

Estrutura de Dados

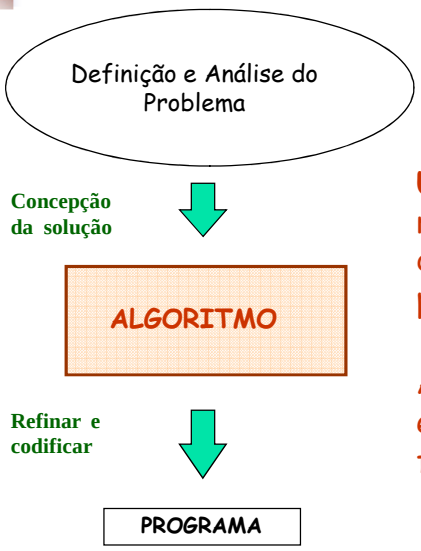
O que é programar?

Programar é

Conceber algoritmos,

expressos numa dada linguagem.

Modelação algorítmica



Um algoritmo é uma representação procedimental de uma solução para um dado problema.

A definição da solução é especificada numa sequência finita de acções.

Algoritmo

Um algoritmo é um conjunto finito e bem definido (não ambíguo) de instruções que descreve os passos lógicos para a realização de uma tarefa.

Um algoritmo correcto é aquele que perante uma entrada válida deve produzir a uma saída única e correcta.

Um algoritmo deve ser eficaz na resolução do problema proposto e eficiente de modo a resolver o problema com o melhor desempenho.

Maria da Conceição Neves

ALGORITMO

- **Algoritmos** – sequência de passos lógicos para a realização de uma tarefa
  - Propriedades
    - **Deve permitir comunicação com o exterior** – Em geral Entrada de Dados e Saída de Resultados
    - **Deve ser finito** -Alcançar a solução em tempo finito
    - **Deve ser bem definido** – sem ambiguidade
- **Concepção e descrição de Algoritmos** – com base na Lógica da Programação concebem-se soluções algorítmicas para os problemas. Os algoritmos são descritos utilizando **Pseudo-Código** ou **Fluxogramas**
- **Há vários métodos de Verificação de Algoritmos, aqui usaremos**
  - **Simulação de execução** (Traçagem) ,
  - **Implementação** (numa **Linguagem de Programação**) e **Testes**

Maria da Conceição Neves

Um algoritmo manipula dados

- Os dados são de variados tipos:

  números (inteiros ou reais), caracteres,

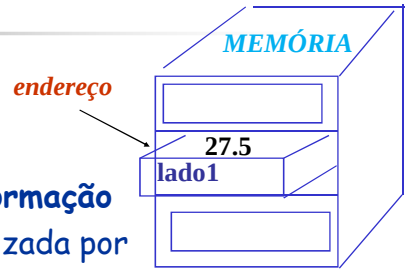
  cadeias de caracteres, valores lógicos, endereços,...

  Cada tipo de dados tem um conjunto de operações associadas

- **Estrutura de Dados** define modo como os dados estão organizados e como são acedidos e alterados .

  Exemplos: variáveis simples,
  arrays mono e multi-dimensionais,
  ficheiros,
  listas, filas, árvores, grafos, ...

Noção de Variável



- **Variável é um Contentor de informação**

É uma posição de memória caracterizada por

  - **nome**
  - **endereço**
  - **valor** (conteúdo)

- **Variável tem associado um Tipo de Dados, que define:**
  - O conjunto de valores que a variável pode armazenar, e
  - O tipo de operações em que as variáveis podem ocorrer

Constantes

- Constantes são variáveis especiais cujo valor é fixo, não se modifica ao longo da execução de um dado programa.
- Num programa em que há valores fixos eles devem ser definidos como constantes
- Exemplos:
  - O valor de ∏ deve ser definido como

    ```
    const real pi=3.1415
    ```
  - A Taxa de IVA para artigos de luxo

    ```
    const real taxaIva=0.21
    ```
  - Maiodade em Portugal

    ```
    const int maioridade=18
    ```


Noção de Estrutura de Dados

A estrutura de dados de um programa é caracterizada pelo modo como se:

- Organiza dos dados em variáveis
- Atribui valores às variáveis
- Acede ao valor das variáveis
- outras operações

Teorema Fundamental da Programação Estruturada

É possível escrever qualquer programa utilizando exclusivamente as três estruturas básicas de controlo:

- **Sequência** - permite a ordenação em série de instruções
- **Selecção/Decisão** - permite a selecção em alternância de um ou outro conjunto de acções por avaliação de uma condição
- **Repetição** - permite a execução condicional em circuito fechado (ciclo) de um dado grupo de instruções. A condição é testada em cada iteração para decidir se sair ou não do ciclo.

Maria da Conceição Neves

O nosso Pseudo-Código

Descrição da Estrutura de Dados **ED** // tipos de dados

Descrição do Processo **ALG** (opcional)
Início
...
Fim

Instruções de Entrada e Saída **ler( )**
escrever( )

Instrução de atribuição **a ← b + c**

Instruções de transferência de controlo de fluxo

Sequência, Decisão/Selecção e Repetição (ver prox.)

Maria da Conceição Neves

O nosso Pseudo-Código (cont.)

Instuções de transferência de controlo de fluxo

Sequência

```
...
...
```

Decisão/Selecção

```
se  ( cond )
        então   ...
        senão    ...
fimse
```

ou

```
se  ( cond )
        então   ...
fimse
```


O nosso Pseudo-Código (cont.)

Instuções de transferência de controlo de fluxo (cont.)

Repetição

```
enquanto  ( cond )  fazer
        ...
fimenquanto
```

ou

```
repete
        ...
enquanto  ( cond )
```

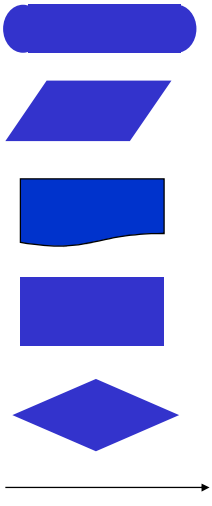
ou

```
para (at_ inic,   c_final,   inc ) fazer
    ...
fimpara
```

Fluxogramas

- Representação Gráfica de um Algoritmo

Símbolos (Versão simplificada)



Inicio / Fim

Entrada (Input)

Saída (Output)

Processos

Decisão

Linhas de Fluxo

Estruturas Fundamentais de Controlo de Fluxo (1)

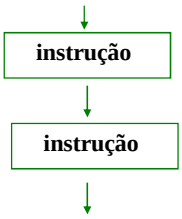
Sequência

Pseudo-Código

instrução

instrução

Fluxograma



(Se necessário separador de instruções ;)

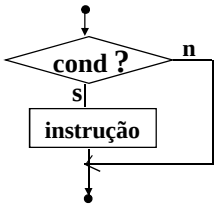
Estruturas Fundamentais de Controlo de Fluxo (2)

Decisão/Selecção

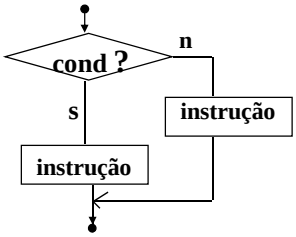
Pseudo-Código

```
se (condição)
      então ...

fimse
```

Fluxograma




```
se (condição)
      então ...

      senão  ...
fimse
```



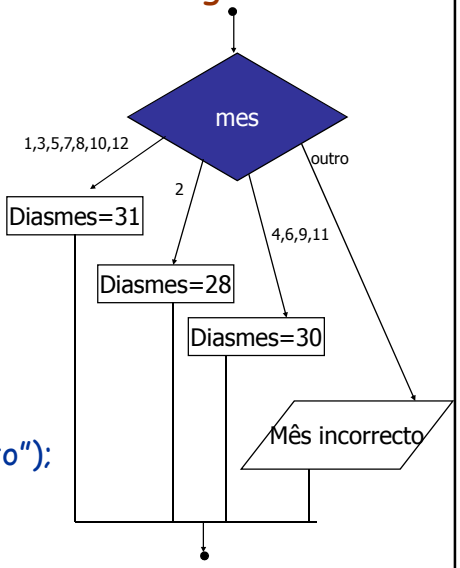
Maria da Conceição Neves

DECISÃO MÚLTIPLA

Pseudo-Código

```
CASO mes SEJA
   1, 3, 5, 7, 8, 10, 12 :
         diasmes=31;
   2:
         diasmes=28;
   4, 6, 9, 11:
         diasmes:30;
  outro:
         escrever("Mês incorrecto");
FIMCASO
```

Fluxograma



E se o ano for bissexto?

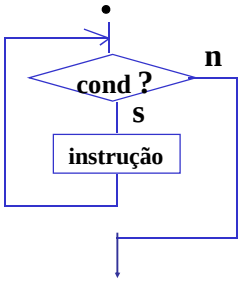
Maria da Conceição Neves

Repetição controlada à entrada

Pseudo-Código

```
enquanto (condição) fazer
        ...
        ...

fimenquanto
```

Fluxograma

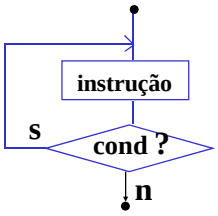


Repetição controlada à saída

Pseudo-Código

```
repetir
  ....

  ...
enquanto (condição)
```

Fluxograma



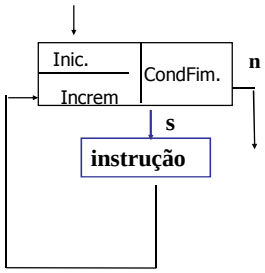
Estruturas Fundamentais de Controlo de Fluxo (3)

Repetição (ciclos for)

Pseudo-Código

```
para ( Inic ; Increm ; CondFim) fazer
    ...
fimpara
```

Fluxograma



Representação da informação em Pseudo-Código

- As linguagens de programação fornecem a possibilidade de representar e manipular vários tipos de informação como por exemplo: inteiros, reais, valores lógicos (Verdadeiro ou Falso), valores alfanuméricos (caracteres e sequências de caracteres).
- Cada Tipo de Dados tem associado um conjunto de operações ou operadores para os manipular.
- Por exemplo:

Tipos de Dados	Operadores Aritméticos
INTEIRO	+ adição - subtracção * multiplicação / divisão inteira (quociente) % resto da divisão
REAL	+ adição - subtracção * multiplicação / divisão (quociente)

EXPRESSÕES

Com operandos e operadores constroem-se expressões

- Vamos organizar os operadores em :
  - OPERADOR DE ATRIBUIÇÃO
  - OPERADORES ARITMÉTICOS
  - OPERADORES RELACIONAIS
  - OPERADORES LÓGICOS

Maria da Conceição Neves

OPERADORES

- OPERADOR DE ATRIBUIÇÃO

  ←
- OPERADORES ARITMÉTICOS

+	-	*	/
adição	subtração	multiplicação	divisão

DIV	MOD
quociente da divisão inteira	resto da divisão inteira
- OPERADORES RELACIONAIS

<	>	<=	>=	=	!=
menor	maior	menor_ou_igual	maior_ou_igual	igual	diferente
- OPERADORES LÓGICOS ou BOOLEANOS

AND	OR	NOT
conjunção	disjunção	negação

Maria da Conceição Neves

Operadores Aritméticos

- Operandos:
  - são números ou expressões numéricas
- Operadores:
  - Adição +
  - Subtracção -
  - Multiplicação *
  - Divisão /
  - Resto da Divisão Inteira MOD
  - Quociente Inteiro DIV
- Tabuadas das Operações
- Propriedades das operações

Noções Básicas de Lógica

- Numa linguagem uma expressão com significado ou é uma **designação** ou uma **afirmação**.
  - **Designação** é uma expressão com significado que designa um objecto

    2 7-3 a
  - **Afirmação** é uma expressão com significado que traduz uma afirmação

    2 > 7 5+1=6
- **Em Lógica considera-se apenas as afirmações sobre as quais se pode dizer se são verdadeiras ou falsas.**

  Estas afirmações designam-se por **proposições**.
- Toda a proposição tem um e um só valor que é **Verdadeiro** (1) ou **Falso**. (0)

Maria da Conceição Neves

Proposições

- Proposições Simples
  - 2+3 = 5 — Valor lógico Verdadeiro
  - 4 é número ímpar — Valor lógico Falso
- Proposições Compostas
  - (2+1 > 2) AND (3 < 5) — Valor lógico Verdadeiro
  - (7 <= 2) OR (3 >2) — Valor lógico Verdadeiro
  - (3+1 = 2) AND (3 >2) — Valor lógico Falso
  - NOT (5>9) — Valor lógico Verdadeiro

Valor lógico Verdadeiro denota-se 1
Valor lógico Falso denota-se 0

Maria da Conceição Neves

Cálculo Proposicional Básico

- Cálculo Proposicional faz o estudo das operações lógicas sobre proposições.
- Operandos são valores lógicos de proposições
- Operações Lógicas ou Booleanas:
  - Conjunção AND
  - Disjunção OR
  - Negação NOT (operador unário)
  - Implicação
  - Equivalência

Operação Negação

- **Operador NOT (operador unário)**
- **Tabela de Verdade:**

Seja p uma proposição

NOT p

p	NOT
0	1
1	0

Se p for falso — **NOT p é verdadeiro**
Se p for verdadeiro — **NOT p é falso**

- **Propriedades**

NOT (NOT p) ⇔ p

Maria da Conceição Neves

Operação Conjunção (AND)

Sejam p, q e r proposições

- Tabela de verdade

P AND q

p \ q	0	1
0	0	0
1	0	1

p AND q é verdadeira se e só se p for verdadeiro e q for verdadeiro.

- Propriedades da operação Conjunção

Comutativa

p AND q ⇔ q AND p

Associativa

(p AND q) AND r ⇔ p AND (q AND r)

Elemento neutro 1 p AND 1 ⇔ 1 AND p ⇔ p

Elemento absorvente 0 p AND 0 ⇔ 0 AND p ⇔ 0

Operação Disjunção (OR)

Sejam p, q e r proposições

- Tabela de verdade

p OR q

p \ q	0	1
0	0	1
1	1	1

***p OR q* é verdadeira se pelo menos uma das proposições o for.**

- Propriedades da operação Disjunção

Comutativa

p OR q ⇔ q OR p

Associativa

(p OR q) OR r ⇔ p OR (q OR r)

Elemento neutro 1 p OR 1 ⇔ 1 OR p ⇔ p

Elemento absorvente 0 p OR 0 ⇔ 0 OR p ⇔ 0

Primeiras Leis de Morgan

Sejam p, q e r proposições

1. A negação da conjunção de proposições tem como consequência a disjunção das proposições negadas

NOT (p AND q) ⇔ (NOT p) OR (NOT q)

2. A negação da disjunção de proposições tem como consequência a conjunção das proposições negadas

NOT (p OR q) ⇔ (NOT p) AND (NOT q)

Maria da Conceição Neves

Expressões com Variáveis - Condições

Chama-se **expressão proposicional ou condição** a qualquer expressão com variáveis que se transforma em proposição (verdadeira ou falsa) sempre que se atribuem valores (dos respectivos domínios) às variáveis

	Condições Lógicas				
	N>0	N<100	NOT(N>0)	(N>0) AND (N<100)	(N>0) OR (N<100)
Se n=45	1	1	0	1	1
Se n=-5	0	1	1	0	1
Se n=105	1	0	0	0	1

Exemplos de Negação de Condições

Condição	Condição negada
N>0	N<=0
(N>0) AND (N<=100)	(N<=0) OR (N>100)
(N>=0) OR (N<100)	(N<0) AND (N>=100)

Maria da Conceição Neves

Algoritmo para Determinação da area de um triângulo

- **ED**

```
var real   base, altura, area
```

- **ALG**

```
INICIO
    ler(base, altura) ;
    area <- base*altura /2  ;
    escrever("Area do Triângulo=", area);
FIM
```

Plano de Testes

Traçagem

Maria da Conceição Neves

Algoritmo para Determinação da área de um rectângulo

- **ED**

  var real lado1, lado2, area

- **ALG**

```
INICIO
    ler(lado1,lado2) ;
    area ← lado1 * lado2  ;
    se (lado1=lado2)
        então escrever("Area Quadrado=", area);
        senão escrever("Area Rectângulo",area);
    fimse
FIM
```

Plano de Testes
Traçagem

Outros Exercícios

1. Descreva um algoritmo que dada a sua idade e a idade do seu amigo determine a relação de idades

```
ED  int minhaIdade, amigoIdade
ALG
Inicio
    ler(minhaIdade, amigoIdade);
    se (minhaIdade= amigoIdade)
        então escrever("São da mesma idade")
        senão  se (minhaIdade > amigoIdade)
                    então  escrever("O seu amigo é mais novo")
                    senão escrever(" O seu amigo é mais velho")
               fimse
    fimse
Fim
```

2. Faça um programa que dadas as medidas dos três lados de um triângulo o classifique quanto aos lados.

Estrururas se decisão encaixadas

EXEMPLO: Faça um programa que dados 3 números inteiros determine o maior e o menor deles.
Obs: (Considere que se os números forem 5,5,5 a resposta Maior 5 e Menor 5 é válida)

E.D. var int n1,n2,n3

```
INICIO
    ler(n1,n2,n3);
    SE (n1>n2)
        ENTAO      /*  n1>n2 */
           SE (n1>n3)
                ENTAO      /* n1>n2 and  n1>n3*/
                    escrever ("O MAIOR é " , n1);
                    SE (n2>n3)
                                ENTÃO /*n1>n2 and  n1>n3 and n2>n3*/
                                escrever ("O MENOR é ", n3);
                       SENÃO     /*n1>n2 and  n1>n3 and n2<=n3*/
                                escrever ("O MENOR é ", n2);
                    FIMSE
                SENÃO      /* n1>n2 and  n1<=n3*/
                    escrever ("O MAIOR é " , n3);
                    escrever ("O MENOR é ", n2);
           FSE
        SENAO      /*  n1<=n2 */
            SE (n2>n3)
                ENTAO     /* n1<=n2 and n2>n3*/
                    escrever ("O MAIOR é " , n2);
                    SE (n1>n3)
                       ENTAO /*n1<=n2 and  n2>n3 and n1>n3*/
                            escrever ("O MENOR é ", n3);
                       SENÃO /*n1<=n2 and  n2>n3 and n1<=n3*/
                            escrever ("O MENOR é ", n1);
                    FIMSE
                SENÃO    /* n1<=n2 and n2<=n3*/
                    escrever ("O MAIOR é " , n3);
                    escrever ("O MENOR é ", n1);
                FIMSE
            FIMSE
FIM
```

Maria da Conceição Neves

Desafio

- Considere que tem oito esferas aparentemente iguais, mas o peso de uma delas (só uma) é diferente do das outras.
  Pretende-se identificar qual a esfera diferente e se ela é mais leve ou mais pesada.
  Para tal dispõe exclusivamente de uma balança de pratos sem pesos e só pode fazer três pesagens.

- Descreva um algoritmo que permita realizar aquela tarefa.

Maria da Conceição Neves

Repetição controlada à entrada

Determinar a média de uma sequência de números terminada por "sentinela"

E.D. **var real** nota, soma, media **var int** contador

```
ALG
INICIO
   contador ← 0                    iniciar contador
   soma ← 0                        iniciar acumulador
   ler (nota)
   enquanto (nota <> sentinela) fazer
      contador ← contador +1       incrementar contador
      soma ← soma +nota            actualizar acumulador
      ler (nota)
   fimenquanto
   media ← soma /contador          Atenção à divisão por 0
   escrever(media)
FIM
```


Exemplo

Faça um programa que dada uma sequência de números, não vazia, terminada por 0 determine qual é o maior número.

```
ED var num, maior
ALG
INICIO
   ler(num)
   maior←num
   ENQUANTO (num<>0) FAZER
         SE (num > maior)
                ENTAO
                maior←num
         FIMSE
           ler(num)
   FIMENQUANTO
   escrever(maior)
FIM
```

Exemplo

Faça um programa que a partir de alguns registos com o número de kms percorridos e da respectiva gasolina gasta determine o consumo médio de gasolina aos 100km.

```
ED var real     km, litros, somakm, somalitros, media
ALG
INICIO
   somakm←0; somalitros←0
   ler(km);
   ENQUANTO (km>0) FAZER
          ler(litros)
           somakm←somakm+km
           somalitros←somalitros+litros
           ler(km)
   FIMENQUANTO;
   media← somalitros/somakm*100
   escrever(media)
FIM
```


Repetição controlada à saída

Determinar a primeira potência de um dado número que é maior do que N

```
E.D.  var int    numero, N, potencia
ALG.
INICIO
   ler(numero)
   ler(N);
   potencia ← 1            iniciar variável
   REPETE
        potencia ← potencia * numero
   ENQUANTO (potencia <= N)
   escrever(potencia)
FIM
```

Exemplo Repetição

Faça um programa permita escrever em sequência os algarismos das unidades, dezenas, ... de um número inteiro. Considere que dispõe das operações div e mod que calculam respectivamente o quociente e o resto da divisão inteira

```
ED var int n,x,y

ALG
INICIO
    ler(n)
    REPETIR
                x ← n div 10
                y ←n mod 10
                escrever (y)
                n ← x
   ENQUANTO (n!=0);
FIM
```

Maria da Conceição Neves

Repetição controlada por contador

Determinar a media da nota de ingresso dos n alunos de uma turma

```
E.D. var real  nota, soma, media var int contador, nalunos
INICIO
   ler(nalunos)
   contador ← 0                    iniciar contador
   soma  ←  0                      iniciar acumulador
   enquanto (contador < nalunos) fazer
       ler (nota)
       contador ← contador +1      incrementar contador
       soma ← soma +nota           actualizar acumulador
   fimenquanto;
   media ← soma /contador
   escrever(media)
FIM
```

Maria da Conceição Neves

Repetição controlada por contador
(com nº de iterações definido)

Determinar a media da nota de ingresso dos n alunos de uma turma

E.D. **var real** nota, soma, media **var int** i, nalunos

```
INICIO
   ler(nalunos)
   soma ← 0                                  iniciar acumulador
   para (i←1 até nalunos passo 1) fazer
      ler (nota)
      soma ← soma +nota                      actualizar acumulador
   fimpara
   media ← soma /nalunos
   escrever(media)
FIM
```


Outros exemplos:

1. Faça um programa que permita escrever a tabuada de um dado número

```
ED  var int num, i
ALG
INICIO
   ler(num)
   PARA( i=1 ATE 10  PASSO 1)
            escrever(num,"x",i ,"=", num*i)
    FIMPARA
FIM
```

2. Faça um programa que permita escrever numa folha A4 as tabuadas do 1 até ao 9.

Obs. Pretende-se formatação usual das tabuadas.

Qual a função do seguinte programa escrito em pseudo-código?

```
E.D.    var  real   Lado1, Area

ALG
Início
  ler (Lado1)      // Instrução de entrada de dados
   Area ← Lado1* Lado1    //Instrução de atribuição
  escrever( "Area do quadrado =", Area);
 /* Instrução de saída de resultados */
Fim
```

Maria da Conceição Neves

Qual a função do seguinte programa escrito em pseudo-código?

```
E.D.    var  real   Lado1, Area
ALG
Início
  ler (Lado1)            /* Instrução de entrada de dados */
  Se ( Lado1<= 0)        /* Instrução de decisão   */
    Então
          escrever( "Quadrado impossível") /*Inst saída  resultados */
    Senão
         Area <-- Lado1* Lado1     /* Instrução de atribuição*/
         escrever( " Area do quadrado =", Area) /*Inst saída result */
  FimSe                  /* Fim de estrutura de decisão*/
Fim
```

Maria da Conceição Neves

Qual a função do seguinte programa escrito em pseudo-código?

```
E.D.    var  real   Lado1, Area
ALG
Início
  ler (Lado1)
  Enquanto ( Lado1>0)    /* Ciclo com teste à entrada */
        Area <-- Lado1* Lado1
        escrever( " Area do quadrado =", Area)
        ler (Lado1)
  FimEnquanto          /* Fim de ciclo*/
Fim
```


Qual a função do seguinte programa escrito em pseudo-código?

```
E.D.    var  int   num, quad
ALG
Início
   Repete          /* Ciclo com teste à saída */
      ler (num)
      quad <-- num* num
      escrever( " O quadrado de", num, "=",quad)
   enquanto (quad < 1000)       /*teste à saída */

Fim
```

Determinação do mínimo múltiplo comum entre dois números

```
ED int  num1, num2, i
Alg
INICIO
        ler(num1, num2)
        SE (num1>num2)
          ENTÃO
                  x=num1; num1=num2; num2=x;
        FIMSE
        i=1;
        ENQUANTO (num2*i   mod  num1  < > 0) FAZER
                          i=i+1
        FIMENQUANTO
        escrever ("O menor múltiplo comum entre", num1, "e",
                  num2 ,"é ", num2*i)
FIM
```

Maria da Conceição Neves

Determinação do máximo divisor comum entre dois números

```
ED int  num1, num2, x, i, flag;
ALG
INICIO
   ler(num1, num2)
   SE (num1=num2)
        ENTÃO          escrever (" O máximo divisor comum é",num1)
        SENÃO
                SE (num1 >  num2)
                      ENTÃO       x=num1; num1=num2; num2=x
             FIMSE
             enc=0
              i=num1
              ENQUANTO (enc=0) FAZER
                    SE (num1 mod i =0)
                      ENTÃO
                            SE (num2 mod i =0)  ENTAO enc=1
                                          SENÃO i= i-1
                            FIMSE
                      SENÃO  i =i -1
                 FIMSE
              FIMENQUANTO
              escrever ("O mdc entre", num1,   "e", num2 ,"é ", i)
   FIMSE
FIM
```

Maria da Conceição Neves

Determinação do máximo divisor comum entre dois números
(Um algoritmo mais eficiente)

```
ED int  num1, num2, x, i;
ALG
INICIO
   ler(num1, num2)
   SE (num1=num2)
        ENTÃO     escrever (" O máximo divisor comum é",num1)
        SENÃO
                SE (num1>num2)
                        ENTÃO
                                x=num1; num1=num2; num2=x
                FIMSE
                i=num1
                ENQUANTO (num2 mod   i  <> 0) FAZER
                        i=i-1
                        ENQUANTO (num1 mod   i  <> 0) FAZER
                                i=i-1
                        FIMENQUANTO
                FIMENQUANTO
                escrever ("O mdc entre",  num1, "e", num2 ,"é ", i)
   FIMSE
FIM
```


A Resolução de Problemas

- Há básicamente dois tipos de problemas:

1. Aqueles para os quais é possível descrever um procedimento determinístico que garante o sucesso.

Só este tipo de problemas é objectivo desta disciplina

2. Aqueles que, não sendo possível descrever um procedimento que garante a solução, exigem uma pesquisa da solução.
A resolução deste tipo de problemas é amplamente estudada no domínio da Inteligência Artificial. Várias estratégias de pesquisa de solução são propostas.
A técnica de "backtracking" é crucial na resolução de muitos problemas deste tipo.

Abstracção

- A descrição de problemas reais inclui um conjunto de detalhes supérfluos. Identificação do problema abstracto.
- A abstracção permite soluções comuns para os problemas que se parecem
- A especificação dos algoritmos e estruturas de dados pode ser realizada em qualquer linguagem. É preciso evitar ambiguidade.
- Precisamos de seleccionar e projectar as estruturas de dados.
- Codificar ou programar é implementar, as estruturas de dados abstractas e os algoritmo, numa linguagem de programação.
- Um algoritmo define um processo. O processo é especificado em termos de instruções simples. Cada passo do processo tem que ser implementado através de uma instrução ou por algum outro algoritmo.
- Algoritmo correcto – produz correctos "outputs" (estados finais) em presença de "inputs" (estados iniciais) válidos

Maria da Conceição Neves

- Dada uma data pretende-se saber quantos dias faltam para o fim desse mês?

Maria da Conceição Neves

- Cálculo das raízes reais de uma equação do 2º dados os seus coeficientes

$$ax^2 + bx + c = 0$$

Maria da Conceição Neves