



Introdução à Programação

Desenvolvimento de Programas

*Licenciaturas em Engenharia Informática,
Engenharia Informática - Pós-laboral e
Engenharia Informática - Curso Europeu*

Resolução de problemas

- Identificar o problema
- Perceber o problema
- Identificar alternativas de resolução
- Escolher a melhor maneira de resolver o problema
- Enumerar as instruções que permitem resolver o problema
- Avaliar a solução obtida

Tipos de Problemas

- Com solução Algorítmica
 - Pode ser descrita como um conjunto de ações
- Com solução Heurística
 - Requer raciocínio baseado em conhecimento ou experiência anterior

Dificuldades na resolução de problemas

- Não definir correctamente o problema
- Não considerar todas as alternativas
- Na avaliação das alternativas, eliminar à partida boas alternativas
- Por desconhecimento do problema escolher alternativas inviáveis
- Não usar uma sequência lógica nos passos da solução
- Ficar preso a pormenores antes de ter uma visão global
- Finalmente, não avaliar correctamente a solução

Desenvolvimento dum programa

PROGRAMAÇÃO TRADICIONAL

Desenvolvimento de um programa =

☞ *codificação*
(programação da solução)

NOVAS METODOLOGIAS

Desenvolvimento de um programa =

- ☞ *Análise do problema*
(Definição do problema, Esboço da solução)
- ☞ *Desenvolvimento da solução*
(Desenvolvimento do algoritmo)
- ☞ *Programação da solução*
(Programação do algoritmo)
- ☞ *Depuração*
(depuração sintáctica, semântica e teste do programa)
- ☞ *Documentação da solução*
- ☞ *Manutenção*

Analisar o problema

- Quais os dados de entrada
- Quais os resultados pretendidos
- Qual o conhecimento requerido pelo problema
- Estratégia
 - Mapa de análise do problema
 - Especificação do problema
 - Outras técnicas

Variáveis e constantes

● Variáveis – servem para guardar informação

■ Simples

- Inteiro
- Real
- Caracter
- ...

Normalmente nomes em letras
minúsculas

(e não contendo +, -, (,), /, *, etc)

■ Estruturadas

- Tabelas
- Ficha
- ...

Exemplo: imposto

● Constantes – tomam valores conhecidos

Normalmente nomes em letras maiúsculas

(e não contendo +, -, (,), /, *, etc)

Exemplo: IMPOSTO

Mapa de Análise do Problema

Dados de entrada	Resultados pretendidos
Dados fornecidos pelo utilizador (ou lidos de ficheiro, ...) necessários à resolução do problema. Informação e formato	Resultados a alcançar com a resolução do problema. Informação e formato
Conhecimento requerido	Estratégia
Conhecimento necessário para a resolução do problema. Fórmulas matemáticas, valores de taxas, ...	Lista de passos a efetuar para solucionar o problema.

Especificação do Problema

- Dados de entrada
 - Dados fornecidos pelo utilizador (ou lidos de ficheiro, ...) necessários à resolução do problema
 - Informação e formato
- Resultados pretendidos
 - Resultados a alcançar com a resolução do problema
 - Informação e formato
- Conhecimento requerido
 - Fórmulas matemáticas, valores de taxas, ...
- Estratégia
 - Lista de passos a efetuar para resolver o problema

Exemplo (1)

Dados de entrada	Resultados pretendidos
horas (inteiro) : número de horas precoHora (real) : preço por hora	pagamentoIliquido (real) : valor sem retirar os descontos
Conhecimento requerido	Estratégia
pagamentoIliquido \leftarrow horas X precoHora	<ul style="list-style-type: none">• Obter do utilizador o número de horas de trabalho e o preço a pagar por hora• Calcular pagamento ilíquido multiplicando o número de horas pelo preço por hora• Mostrar valor resultante ao utilizador

(ou) Exemplo (1)

Considerando preço por hora constante

- Dados de entrada
 - horas (inteiro) : número de horas
- Resultados pretendidos
 - pagamento líquido (real) : valor a pagar sem retirar descontos
- Conhecimento requerido
 - $\text{PRECOHORA} = 7.5$: preço por hora (em €)
 - $\text{pagamento líquido} \leftarrow \text{horas} \times \text{PRECOHORA}$
- Estratégia
 - Obter do utilizador o número de horas de trabalho
 - Calcular pagamento líquido multiplicando o número de horas pelo preço por hora
 - Mostrar valor resultante ao utilizador

Desenvolvimento da solução

- Solução algorítmica:
 - Fluxograma (*Flowchart*)
 - Pseudocódigo
 - Outras técnicas




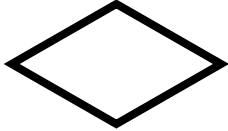

Solução algorítmica

- Um algoritmo é constituído por um conjunto de instruções
- As instruções são executadas sequencialmente
- Numa primeira abordagem podemos distinguir três tipos diferentes de instruções:
 - Entradas/Saídas
 - Ações
 - Instruções que alteram o fluxo do programa:
 - Decisões
 - Ciclos (ou repetições)
 - Chamadas a Funções

Algoritmos – Fluxogramas (1)

- Esquematizam o fluir da execução das instruções por parte do computador
- Podem ser utilizados para descrever programas inteiros
- Possuem uma grande correspondência com a sequência de instruções executadas pelo computador

Algoritmos - Fluxogramas (2)

- Linhas de ligação 
- Inicio/Fim 
- Ações / Processamento 
- Decisão 
- Entradas/Saídas 

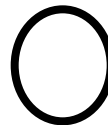
Algoritmos - Fluxogramas (3)

- Módulos (funções/ procedimentos)

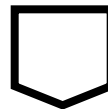


- Conectores

- na mesma página

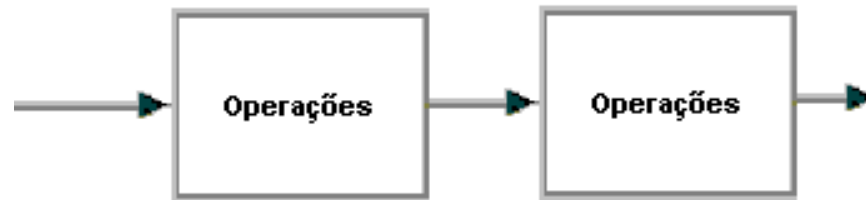


- para outra página

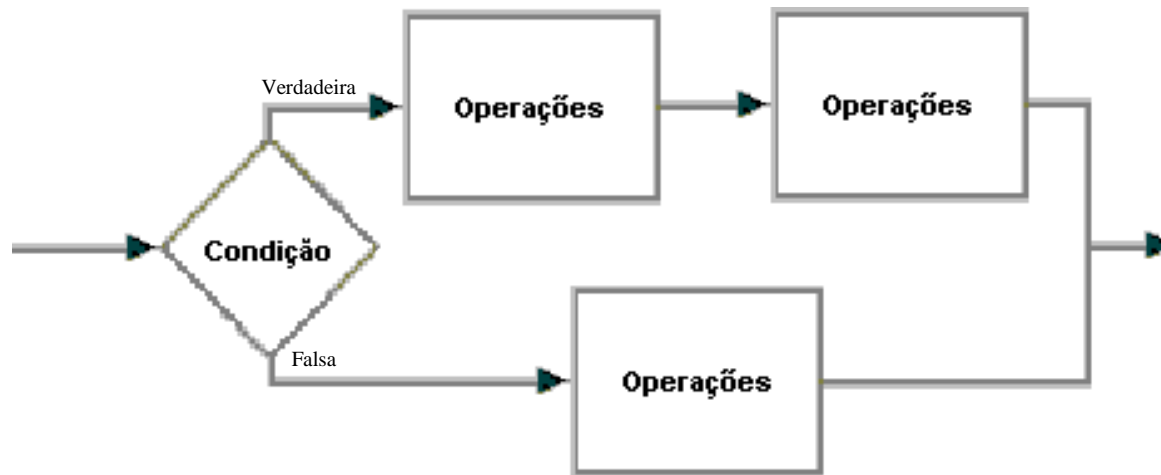


Algoritmos - Fluxogramas (4)

- Sequência (ou bloco de instruções)

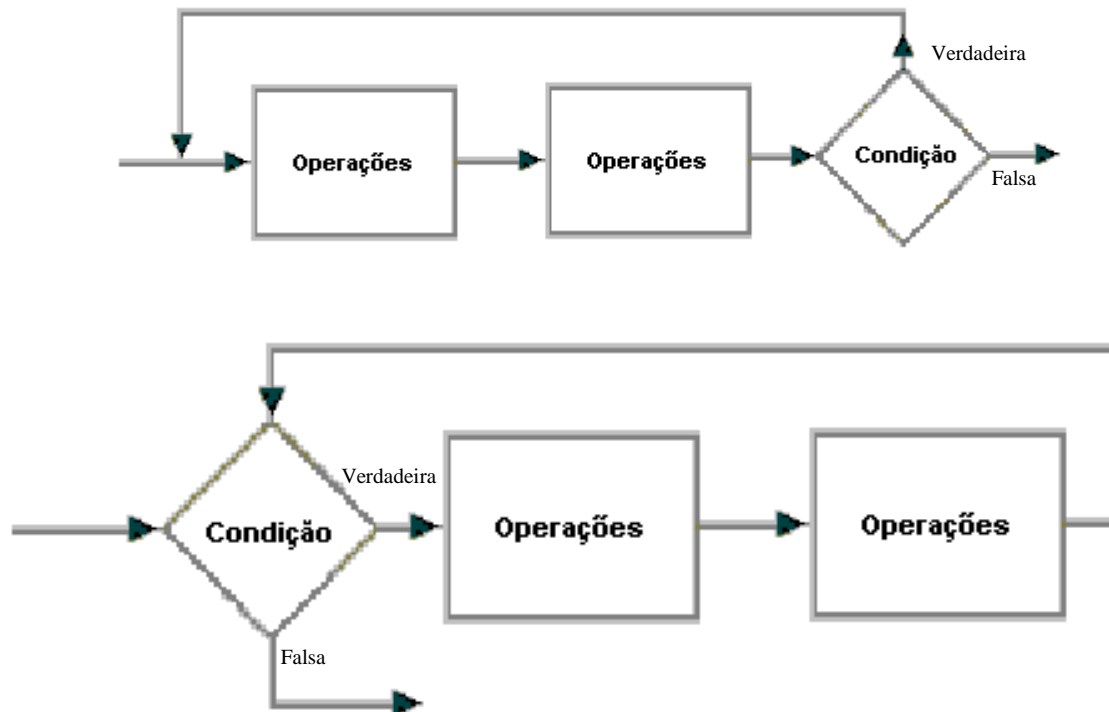


- Decisão



Algoritmos - Fluxogramas (5)

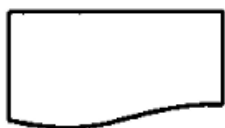
● Ciclos



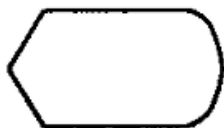
Permitem representar qualquer tipo de ciclo

Algoritmos - Fluxogramas (6)

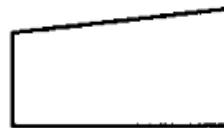
Símbolos especializados de entrada e saída



**Documento
em
impressora**



Ecrã



Teclado



**Disco
magnético**



**Tambor
magnético**



**Cartão
perfurado**



**Fita
magnética**

Algoritmos - Fluxogramas (7)

- Pontos fortes
 - Compreensão universal e imediata
 - Aproximação das linguagens de programação

- Pontos fracos
 - Dizem muito pouco acerca das estruturas de dados
 - Nível de abstracção muito baixo
 - Facilmente perdem o sincronismo com a análise
 - Qualquer alteração obriga a re-desenhar o fluxograma

Algoritmos - Pseudocódigo (1)

- Correspondem à utilização da semântica das linguagens de programação em linguagem corrente
- Conjunto limitado de instruções lógicas e condicionais com substantivos e verbos fortes
- Pretendem exprimir:
 - Ações primitivas
 - Informam sobre algo que deve ser feito
 - Estruturas de controlo
 - Sequência
 - Seleção
 - Repetição

Algoritmos - Pseudocódigo (2)

● Definição de bloco de instruções

INÍCIO NomeBloco

instrução

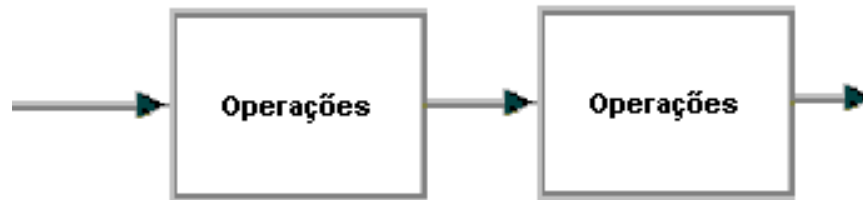
instrução

.

.

.

FIM NomeBloco



Algoritmos - Pseudocódigo (3)

- Entrada / Saída
 - OBTÉM (<var>)
 - MOSTRA (<var>)
- Leitura / Escrita para disco
 - LÊ (<var>)
 - ESCREVE (<var>)

Algoritmos - Pseudocódigo (4)

● Decisão

SE <condição>

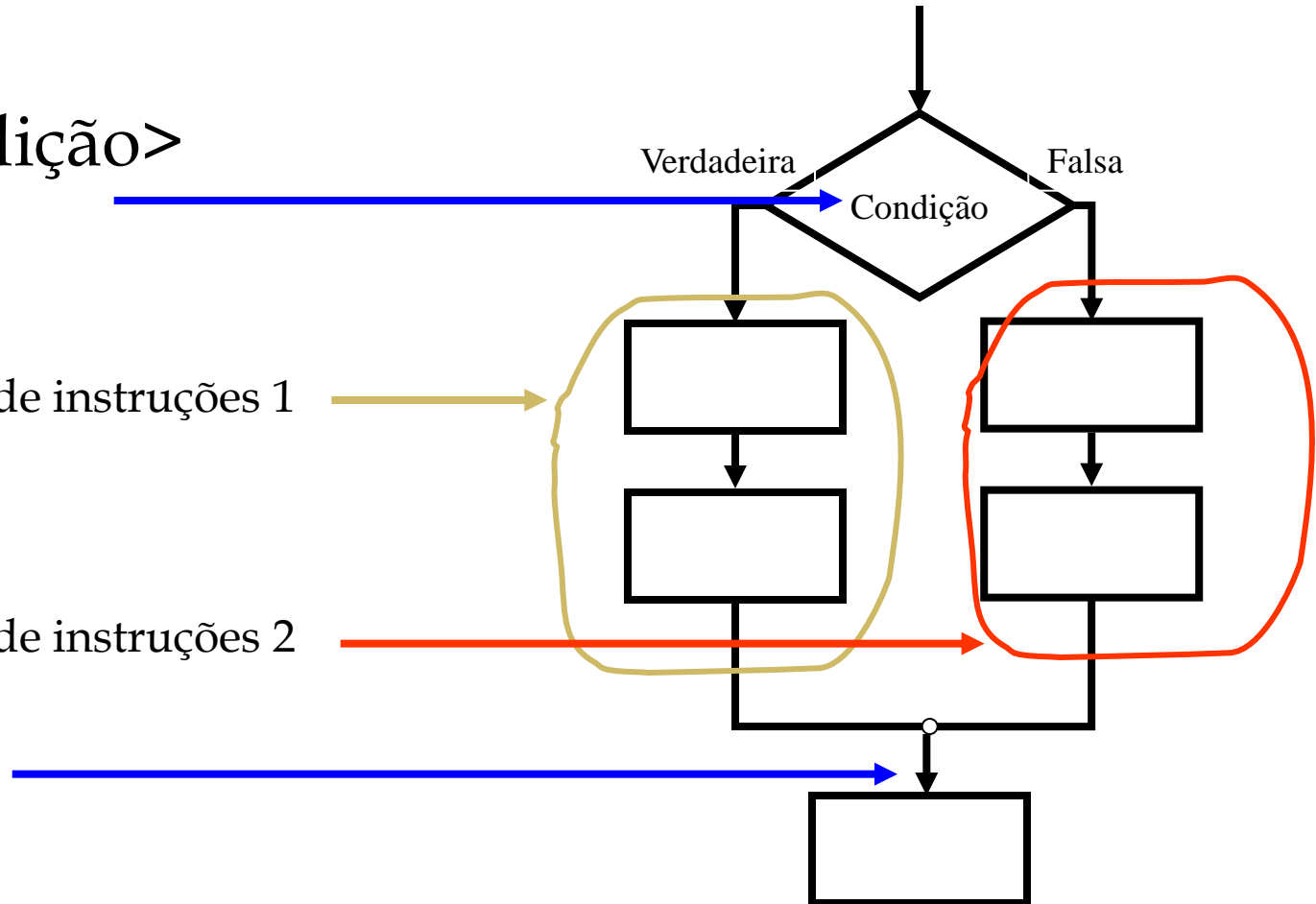
ENTÃO

bloco de instruções 1

SENÃO

bloco de instruções 2

FIM SE



Algoritmos - Pseudocódigo (5)

● Decisão

CASO <var>

QUANDO < valor 1>

bloco de instruções 1

QUANDO <valor 2>

bloco de instruções 2

...

FIM CASO

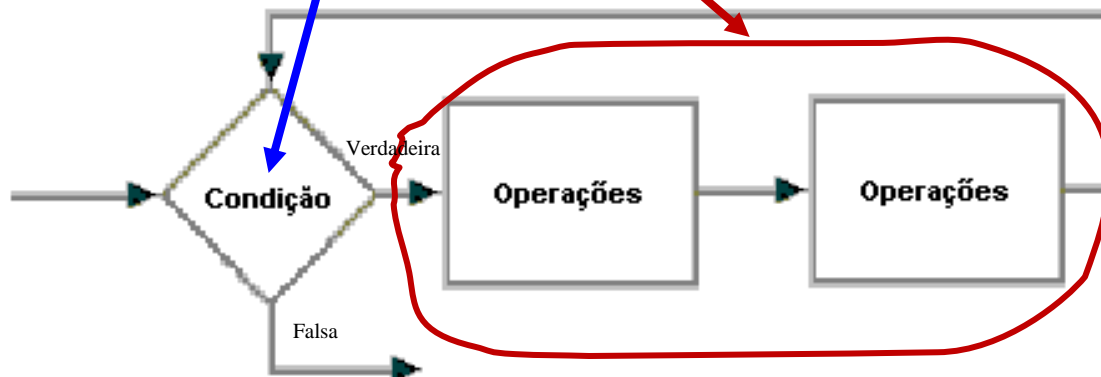
Algoritmos - Pseudocódigo (6)

● Ciclos

ENQUANTO <condição> FAZER

Bloco de instruções

FIM ENQUANTO



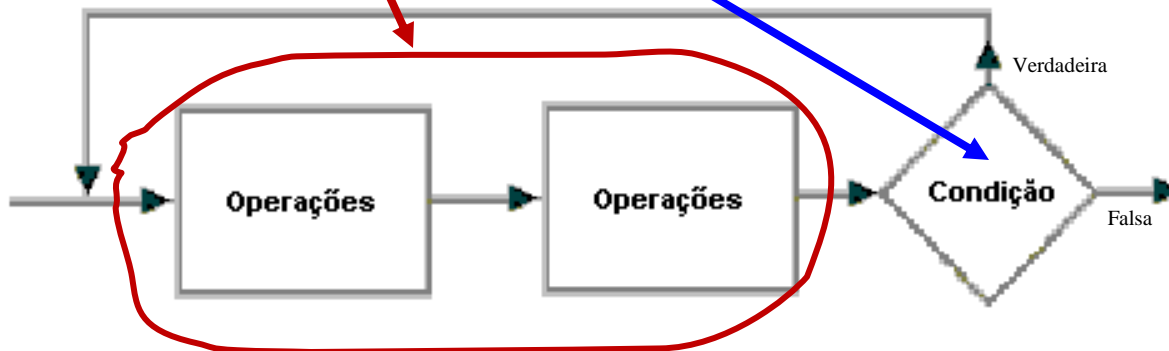
Algoritmos - Pseudocódigo (7)

● Ciclos

FAZER

Bloco de instruções

ENQUANTO <condição>



Algoritmos - Pseudocódigo (8)

- Chamada de função/ procedimento
 - CHAMA <NomeFunção>(<var1,var2,...>)
 - <var3 > ← CHAMA <NomeFunção> (<var1, var2,...>)
 - CHAMA <NomeFunção> ()
 - <var3 > ← CHAMA <NomeFunção> ()

Algoritmos - Pseudocódigo (10)

● Outros símbolos

- $\langle \dots \rangle$ substituir
- $[\dots | \dots]$ escolha entre vários
- $\{ \dots \}$ repetição

Algoritmos - Pseudocódigo (11)

- Pontos fortes
 - Versatilidade
 - Elaboração rápida e fácil

- Pontos fracos
 - Distanciamento das linguagens de programação
 - Inexistência de um padrão
 - Pode conduzir a falta de especificidade
 - Pode criar alguma ambiguidade

Exemplo (1)

Por intermédio de pseudocódigo:

INÍCIO CalculaPagamentoIliquido1

OBTÉM (horas, precoHora)

 pagamentoIliquido \leftarrow horas X precoHora

MOSTRA (pagamentoIliquido)

FIM CalculaPagamentoIliquido1

Por intermédio de fluxograma:

Feito como exemplo na aula!

Exemplo (2)

Considerar que o funcionário pode trabalhar em horário normal ou pós-laboral (as horas pós-laborais são pagas a 150%)

Mapa de Análise do Problema

Dados de entrada	Resultados pretendidos
horas (inteiro) : numero de horas tipoHorario (caracter) : 'N' (ou 'n') se horário pós-laboral, outro caracter se horário normal precoHora (real): preço por hora normal	pagamento (real) : valor sem retirar os descontos
Conhecimento requerido	Estratégia
TAXA_PL=1.5: taxa do trabalho pós-laboral pagamento \leftarrow horas X precoHora ou pagamento \leftarrow horas X precoHora X TAXA_PL	<ul style="list-style-type: none">• Obter do utilizador o nº de horas de trabalho e o tipo de horário (normal ou pós-laboral)• Calcular pagamento multiplicando o nº de horas pelo preço por hora e, no caso de ser trabalho pós-laboral, multiplicar ainda pela respetiva taxa• Mostrar valor calculado ao utilizador

Exemplo (2)

Por intermédio de pseudocódigo:

INÍCIO CalculaPagamentoIliquido2

OBTÉM (horas, precoHora)

OBTÉM (tipoHorario)

SE (tipoHorario= 'n' ou tipoHorario= 'N')

ENTÃO

 pagamento \leftarrow horas X precoHora X TAXA_PL

SENÃO

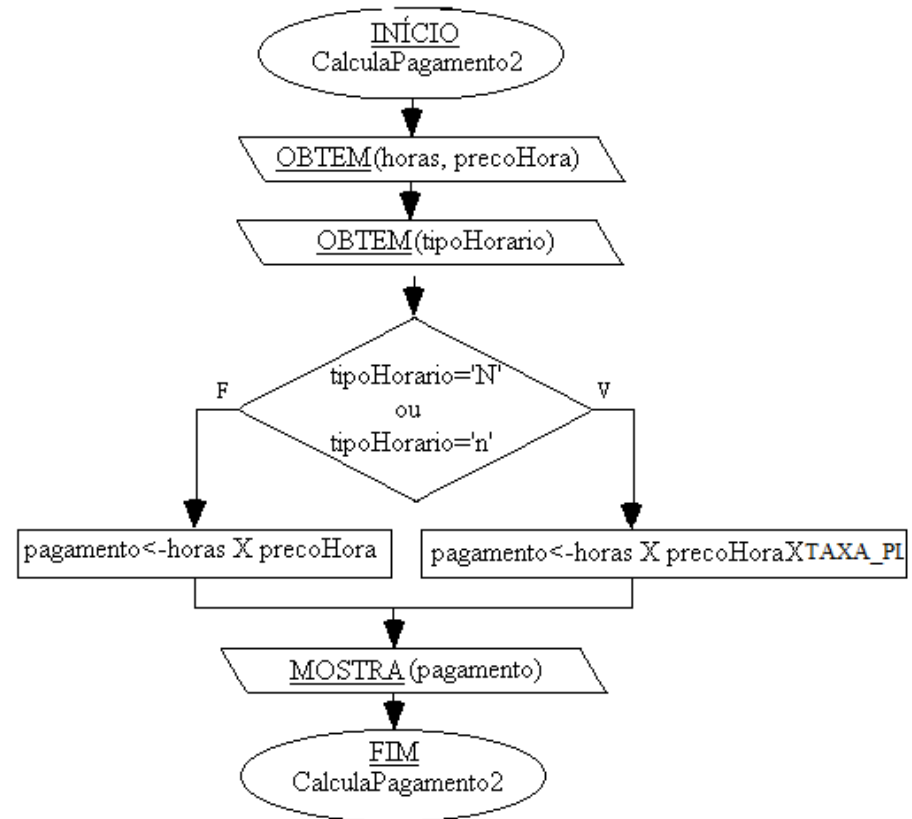
 pagamento \leftarrow horas X precoHora

FIM SE

MOSTRA (pagamento)

FIM CalculaPagamentoIliquido2

Por intermédio de fluxograma:



Exemplo (3)

Calcular vencimento do funcionário durante 1 semana (7 dias)

Mapa de Análise do Problema

Dados de entrada	Resultados pretendidos
horas (inteiro) : numero de horas; tipoHorario (caracter) : 'N' (ou 'n') se horário pós-laboral, outro caracter se horário normal; precoHora (real): preço por hora.	pagamento (real) : valor sem retirar os descontos.
Conhecimento requerido	Estratégia
TAXA_PL=1.5: taxa do trabalho pós- laboral pagamento \leftarrow horas X precoHora ou pagamento \leftarrow horas X precoHora X TAXA_PL	<ul style="list-style-type: none">• Obter n° de horas, preço por hora e tipo de horário• Para cada dia da semana calcular o pagamento diário (multiplicando o n° de horas pelo preço por hora e pela taxa de 150% caso seja trabalho pós-laboral), acumulando o resultado no valor do pagamento semanal• Mostrar pagamento semanal

(ou) Exemplo (3)

Calcular vencimento do funcionário durante 1 semana (7 dias)

Especificação do problema

● Dados de entrada

- horas (inteiro) : numero de horas;
- tipoHorario (character) : 'N' (ou 'n') se horário pós-laboral, outro character se horário normal;
- precoHora (real) : preço por hora

● Resultados pretendidos

- pagamento (real) : valor a pagar sem retirar descontos

● Conhecimento requerido

- TAXA_PL=1.5 : taxa do trabalho pós-laboral
- pagamento \leftarrow horas X precoHora
- pagamento \leftarrow horas X precoHora X TAXA_PL

● Estratégia

- Para cada dia da semana calcular pagamento diário (multiplicando o n° de horas trabalhadas pelo preço por hora e, no caso do trabalho ser pós-laboral, pela taxa de 150%), acumulando o resultado no valor do pagamento semanal
- Mostrar pagamento semanal

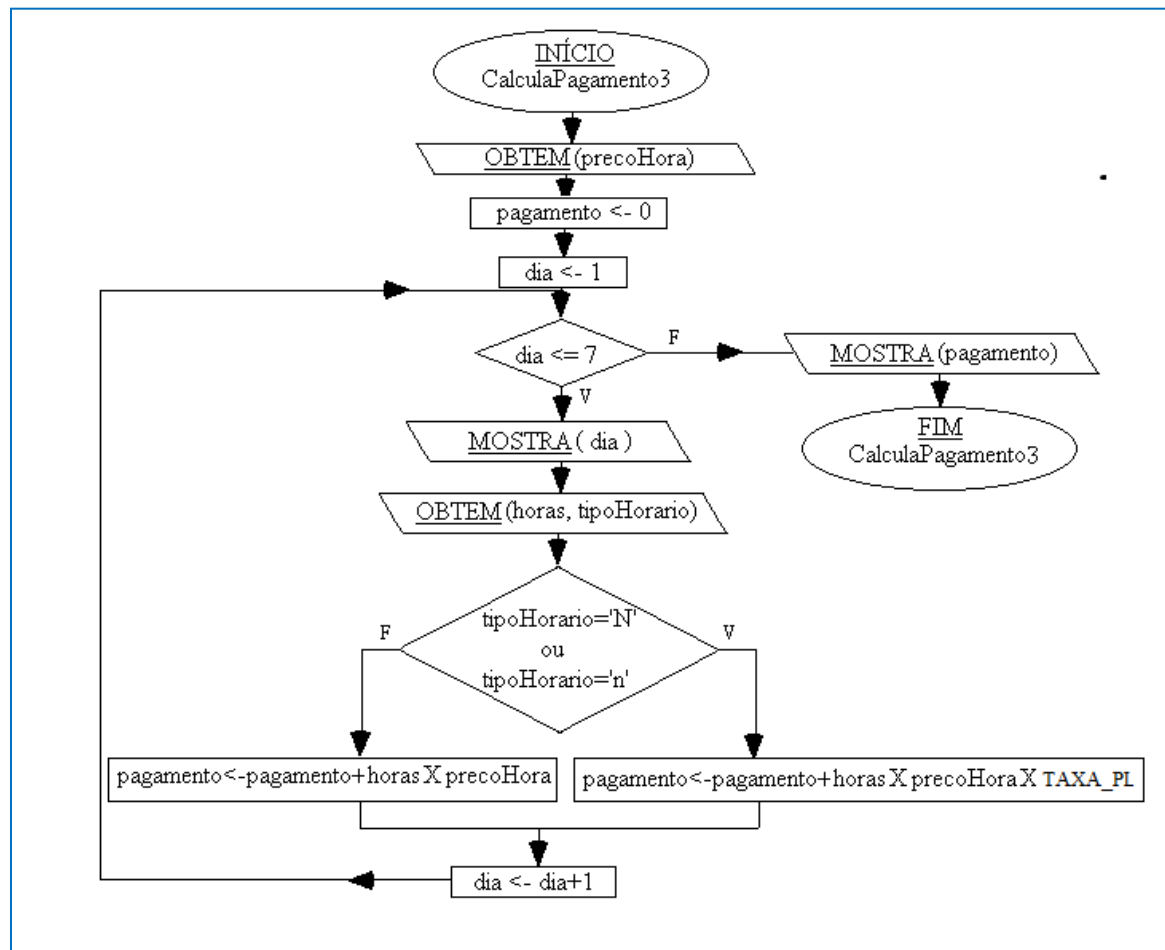
Exemplo (3) (7 dias da semana)

Por intermédio de pseudocódigo:

```
INÍCIO CalculaPagamentoIliquido3
  OBTEM (precoHora)
  pagamento ← 0
  dia ← 1
  ENQUANTO (dia ≤ 7) FAZER
    MOSTRA (dia)
    OBTEM (horas, tipoHorario)
    SE (tipoHorario = 'n' ou tipoHorario = 'N')
      ENTÃO
        pagamento ← pagamento + horas X precoHora X TAXA_PL
      SENÃO
        pagamento ← pagamento + horas X precoHora
    FIM SE
    dia ← dia + 1
  FIM ENQUANTO
  MOSTRA (pagamento)
FIM CalculaPagamentoIliquido3
```

Exemplo (3) (7 dias da semana)

Por intermédio de fluxograma:



Funções

- Conjunto de instruções que possibilitam executar uma ação mais específica sobre um conjunto de dados de modo a obter um conjunto de resultados
- Aparecem agrupadas sob um nome

Normalmente têm nomes elucidativos sobre as acções que executam, com a 1ª letra de cada “palavra” em maiúsculas e as restantes em minúsculas (não contendo +, -, (,), /, *, etc.)

Exemplo: **CalculaImposto**

Programa que calcula salário líquido

Programa principal (exemplo (1))

Pseudocódigo

INÍCIO Programa

(horas, precoHora) ← CHAMA ObtemHPrecoH()

pagamentoIlíquido ← CHAMA Calcula (horas, precoHoras)

CHAMA MostraPaga (pagamentoIlíquido)

FIM Programa

INÍCIO Calcula

RECEBE (horas, pH)

Pagamento ← horas X pH

DEVOLVE (pagamento)

FIM Calcula

INÍCIO ObtemHPrecoH

RECEBE()

OBTÉM (horas)

OBTÉM (pHora)

DEVOLVE(horas, pHora)

FIM ObtemHPrecoH

INÍCIO MostraPaga

RECEBE (pagamento)

MOSTRA
(pagamento)

DEVOLVE ()

FIM MostraPaga

Algoritmos - Pseudocódigo

● Chamada de função/ procedimento

- CHAMA <NomeFunção>(<var1,var2,...>)
- <var3 > ← CHAMA <NomeFunção> (<var1, var2,...>) ...
- CHAMA <NomeFunção> ()
- <var3 > ← CHAMA <NomeFunção> ()

INÍCIO <NomeFuncao>

RECEBE(<var1, var2, ...>)

Instruções

...

DEVOLVE(<var3, var4, ...>)

FIM <NomeFuncao>

Exemplos de aplicação das técnicas (com funções)

Calculo das raízes de um polinómio de 2º grau (1)

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

Mapa de Análise do Problema

Dados de entrada	Resultados pretendidos
a,b,c (reais): coeficientes do polinómio de 2º grau	Raízes do polinómio, reais ou imaginárias
Conhecimento requerido	Estratégia
$x = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}$ $i = \sqrt{-1}$	<ul style="list-style-type: none">• Determina o valor de $(b^2 - 4ac)$• Se for negativo, calcula raízes imaginárias (chamando função respetiva)• Senão, calcula raízes reais (chamando função respetiva)

Exemplos de aplicação das técnicas (com funções)

Calculo das raízes de um polinómio do 2º grau (2)

Pseudocódigo

Encontre erros nos algoritmos

FAZER

MOSTRA “introduza os coeficientes da eq. 2º grau “

RECEBE “A,B,C”

SE $B^2 < 4AC$

ENTÃO

Calcula RR (A,B,C)

SENÃO

Calcula RI (A,B,C)

FIM SE

MOSTRA “Quer continuar (s/n) ?”

RECEBE CONT

ENQUANTO (CONT = s)

INÍCIO RAIZES

FAZER

MOSTRA (“Introduza os coeficientes da eq. 2º grau “)

OBTÉM (a,b,c)

SE ($b^2 < 4ac$)

ENTÃO

CHAMA Calcula RI (a,b,c)

SENÃO

CHAMA Calcula RR (a,b,c)

FIM SE

MOSTRA (“Quer continuar (s/n) ?”)

OBTÉM (cont)

ENQUANTO (cont = ‘s’ ou cont = ‘S’)

FIM RAIZES

Mais exemplos de aplicação das técnicas (com funções)

Pseudocódigo

No caso das funções usa-se explicitamente a palavra “Função”!

```
INÍCIO Principal
  n ← -6
  FAZER
    a ← CHAMA F1(n)
    MOSTRA (a)
    (a,n) ← CHAMA F2(a)
    MOSTRA (a, n)
  ENQUANTO(n>0)
  MOSTRA (n)
  CHAMA F3(n, a)
FIM Principal
```

```
INÍCIO FUNÇÃO F1
  RECEBE (i)
    i ← i-1
  DEVOLVE (i)
FIM FUNÇÃO F1
```

```
INÍCIO FUNÇÃO F2
  RECEBE (i)
    x ← i X i
    y ← i-1
    MOSTRA (x)
  DEVOLVE (x,y)
FIM FUNÇÃO F2
```

```
INÍCIO FUNÇÃO F3
  RECEBE (y,z)
    y ← y-z
    MOSTRA (y)
  DEVOLVE ()
FIM FUNÇÃO F3
```

Exemplos de aplicação das técnicas (com funções)

Determinar se dois números são primos entre si (1)

Mapa de Analise do Problema

Dados de entrada	Resultados pretendidos
a, b (inteiros): dois números à escolha do utilizador	Mensagem a indicar se números são, ou não, primos entre si
Conhecimento requerido	Estratégia
Dois números são primos entre si se o único divisor comum for a unidade	<ul style="list-style-type: none">• Verifica se existe algum divisor comum a <u>a</u> e <u>b</u> para além da unidade• Se não existir, informa que números são primos entre si• Senão, informa que números não são primos entre si

Exemplos de aplicação das técnicas (com funções)

Determinar se dois números são primos entre si (2)

Pseudocódigo

```
INÍCIO Primos
  OBTÉM(a, b)
  p ← 'S'
  i ← 2
  ENQUANTO (p='S' E i ≤ a)
    SE (RESTO (a // i) = 0) ENTÃO // → Divisão Inteira
      SE (RESTO (b // i) = 0) ENTÃO
        p ← 'N'
      FIM SE
    FIM SE
    i ← i+1
  FIM ENQUANTO
  SE (p='S') ENTÃO
    MOSTRA ("Os números são primos entre si")
  SENÃO
    MOSTRA ("Os números não são primos entre si")
  FIM SE
FIM Primos
```

Exemplos de aplicação das técnicas (com funções)

Determinar se dois números são primos entre si (3)

INÍCIO Primos

OBTÉM(a, b)

SE (a < b) ENTÃO

p ← CHAMA Pr(a, b)

SENÃO

p ← CHAMA Pr(b, a)

FIM SE

SE (p = 'S') ENTÃO

MOSTRA (" Os números são primos entre si")

SENÃO

MOSTRA ("Os números não são primos entre si")

FIM SE

FIM Primos

INÍCIO Pr

RECEBE(menor, maior)

p ← 'S'

i ← 2

ENQUANTO (p='S' e i ≤ menor)

SE (RESTO (menor // i) = 0) ENTÃO

SE (RESTO (maior // i)=0) ENTÃO

p ← 'N'

FIM SE

FIM SE

i ← i+1

FIM ENQUANTO

DEVOLVE (p)

FIM Pr

// → Divisão Inteira

Decisões Encadeadas

- Tipos de lógicas
 - Lógica independente (Straight-Through)
 - Cada decisão é independente da decisão seguinte
Normalmente não é utilizado o bloco correspondente ao SENÃO (else)
 - Lógica Positiva
 - No caso da condição ser verdadeira (ENTÃO) é executada uma ação
 - No caso da condição ser falsa (SENÃO) é executada outra condição
 - Lógica Negativa
 - No caso da condição ser falsa (SENÃO) é executada uma acção
 - No caso da condição ser verdadeira (ENTÃO) é executada outra condição

Lógica Independente

...

SE (idade < 16)

ENTÃO

custo \leftarrow 7

FIM SE

SE (idade \geq 16 e idade < 65)

ENTÃO

custo \leftarrow 10

FIM SE

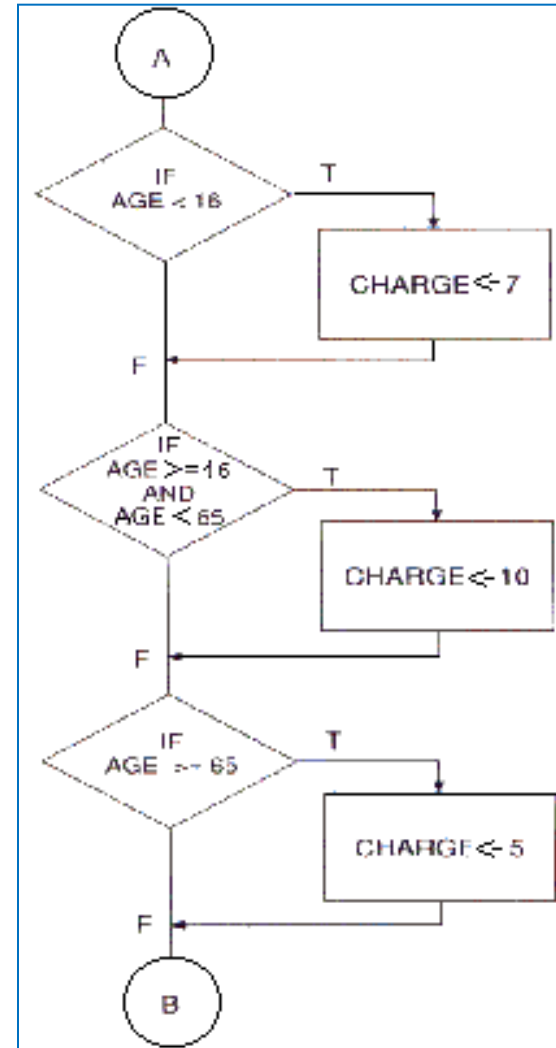
SE (idade \geq 65)

ENTÃO

custo \leftarrow 5

FIM SE

...



Lógica Positiva

...

SE (idade < 16)

ENTÃO

custo \leftarrow 7

SENÃO

SE (idade < 65)

ENTÃO

custo \leftarrow 10

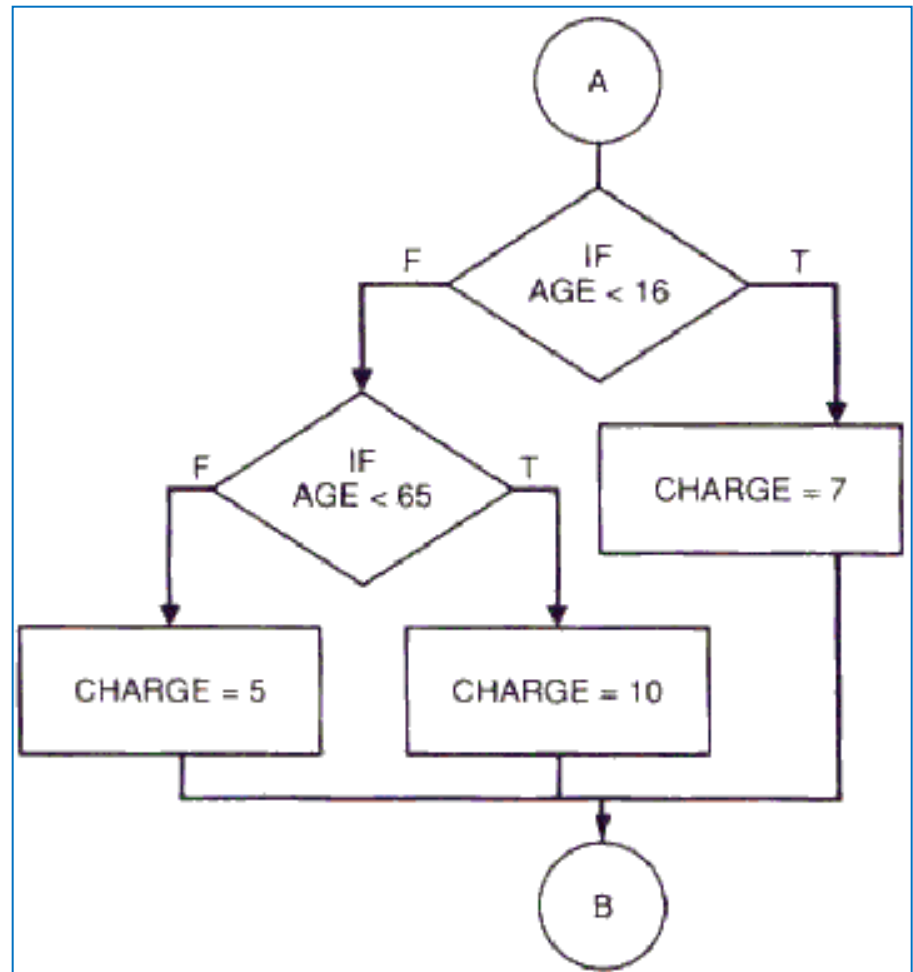
SENÃO

custo \leftarrow 5

FIM SE

FIM SE

...



Lógica Negativa

...

SE (idade ≥ 16)

ENTÃO

SE (idade ≥ 65)

ENTÃO

custo $\leftarrow 5$

SENÃO

custo $\leftarrow 10$

FIM SE

SENÃO

custo $\leftarrow 7$

FIM SE

...

