



INSTITUTO DO EMPREGO E FORMAÇÃO PROFISSIONAL, IP

Centro de Emprego e Formação Profissional de Aveiro

Serviço de Formação Profissional de Aveiro

Algoritmia

Conceitos introdutórios

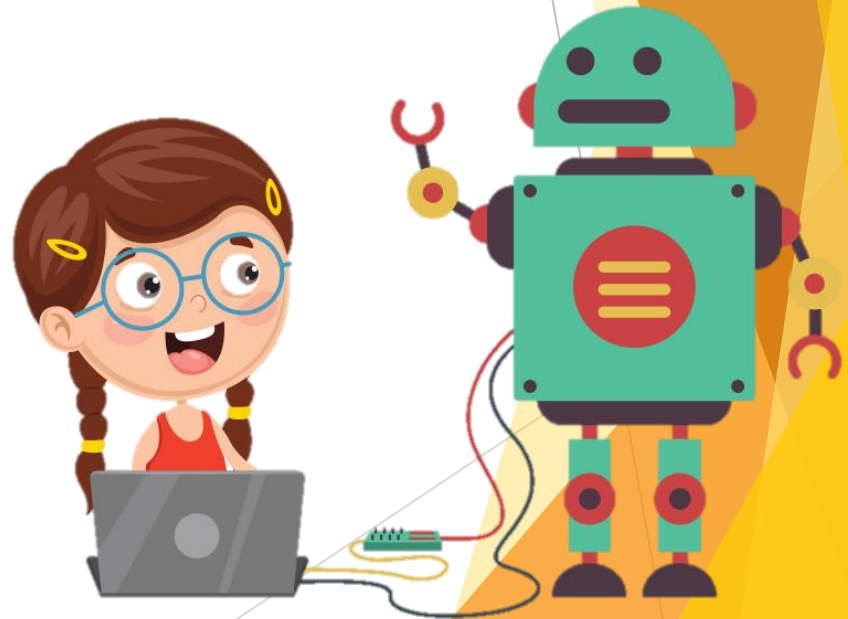
Programas - o que são?

Os computadores são burros! Se não lhes “ensinarmos” como devem proceder para realizar as suas tarefas, eles não serão capazes de as realizar sozinhos!

São os humanos que dão a inteligência aos computadores, **programando-os!**

Um **programa informático**, trata-se na sua essência, de um conjunto de instruções (ou comandos) sequenciais, destinados a indicar como um (ou mais) sistemas computacionais deverão realizar as suas tarefas ou funções.

Imagine um programa informático como um guia passo-a-passo de como um computador deverá realizar uma determinada tarefa.



Programas - o que são?

Imaginemos o computador como sendo uma pessoa sem qualquer conhecimento do mundo, mas com uma capacidade extraordinária de memorizar, fazer comparações e cálculos matemáticos.

Descubra as 3 diferenças



Imaginemos ainda darmos-lhe este problema sem ele ter qualquer tipo de conhecimento do jogo das diferenças.

O computador não saberia o que fazer, porque não estava instruído ou teria alguma ideia de qual o problema e quais as hipóteses para a sua resolução.

No entanto, se o ensinarmos, ele não só passará a conseguir resolver este jogo das diferenças, como qualquer outro que siga o mesmo formato - o computador passará a ter conhecimento de como resolver o jogo das diferenças!

Programas - o que são?

O que teríamos que dizer ao computador para ele conseguir resolver o jogo das diferenças?

Descubra as 3 diferenças



1. Sobrepor as duas imagens (cálculos matemáticos).
2. Analisar grupos distintos de cor (comparar).
3. Marcar os grupos com um círculo (memorizar).
4. Apresentar os resultados.



Algoritmos

Exercício 1

Descubra as 7 diferenças



Algoritmos

Exercício 1 (solução)



Programas - linguagens de programação

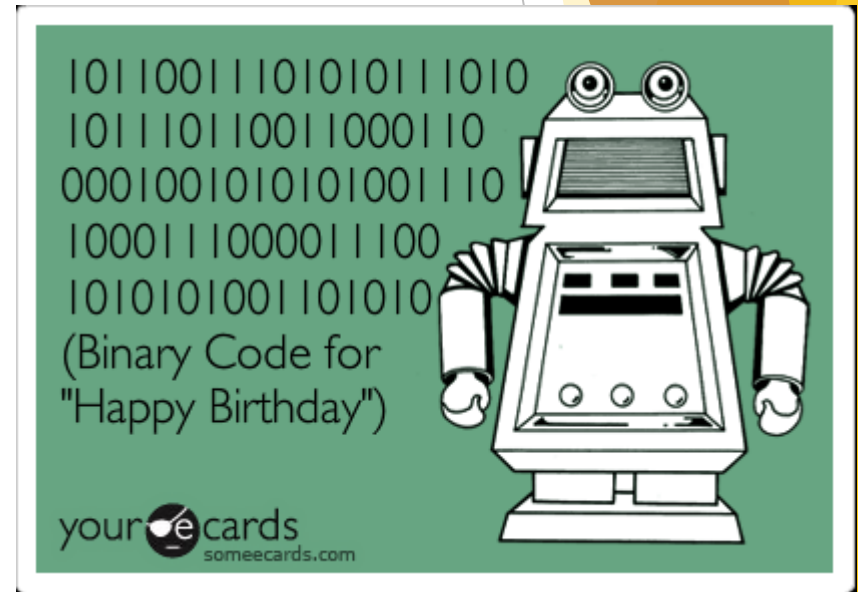
Assim como nós humanos utilizamos uma linguagem (por exemplo: o português) para comunicarmos e darmos instruções uns aos outros, os computadores também têm a capacidade de interpretar a sua própria linguagem (textualmente), que no caso não será a mesma que a dos humanos! Desta forma, mandarmos um computador realizar uma instrução, falando em português com ele, não será o caminho certo a seguir! Precisamos de falar a linguagem dele!

Os computadores entendem **linguagem máquina** (também conhecida como **linguagem de programação de baixo-nível**), que em termos gerais, nada mais são do que conjuntos de sequencias binárias com um determinado sentido lógico, compreendidos pelo computador.



Programas - linguagens de programação

No mesmo sentido que nós humanos entendemos sequencias de letras, que dispostas sequencialmente formulam palavras e por sua vez frases com um determinado sentido, no caso dos computadores, estes entendem bits (0 e 1) que dispostos sequencialmente formulam comandos, e por sua vez instruções com um determinado sentido.



Programas - linguagens de programação

Mas visto que se tornaria muito confuso para nós humanos “comunicar” com os computadores em binário, decidiu-se criar uma forma mais fácil para o realizar, dando-se assim ao desenvolvimento de **linguagens de programação de alto-nível**.

Uma **linguagens de programação de alto-nível**, trata-se de um intermédio entre a linguagem humana e a linguagem máquina, com termos por nós humanos compreendidos.

Vejamos assim as linguagens de alto-nível como o sistema intermédio, que levará o computador a entender as instruções que lhe foram ditadas pelo humano.



```
1  program Hello;  
2  begin  
3      writeln ('Hello, world.');
```

}

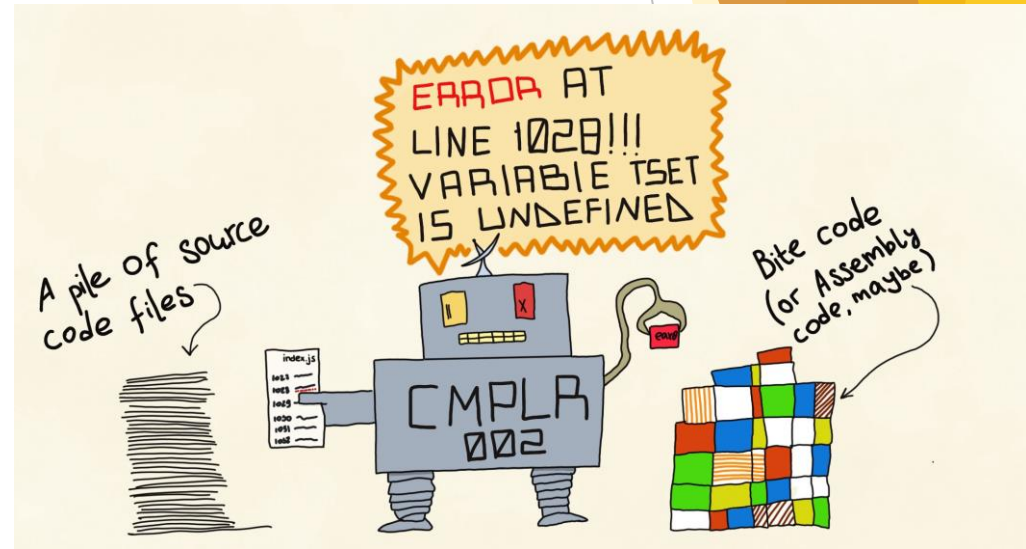
```
4      readln;  
5  end.
```

Programas - compiladores

No entanto, a compreensão entre homem e máquina, tendo por base **uma linguagem de alto nível não é direta!**

Uma linguagem de alto-nível é-nos mais facilmente compreendida, do que será por um computador! Isto porque as linguagens de alto-nível foram desenvolvidas para facilitar a vida aos humanos e não às máquinas!

Neste sentido, nasceram os **compiladores**, que nada mais são do que tradutores. Estes “pegam” no que nós humanos escrevemos (numa linguagem de alto nível), e traduzem tudo para linguagem-máquina - para que o computador possa então entender o que lhe pretende ser incumbido.



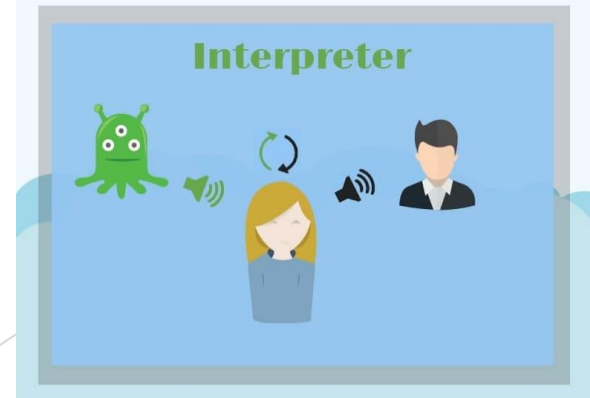
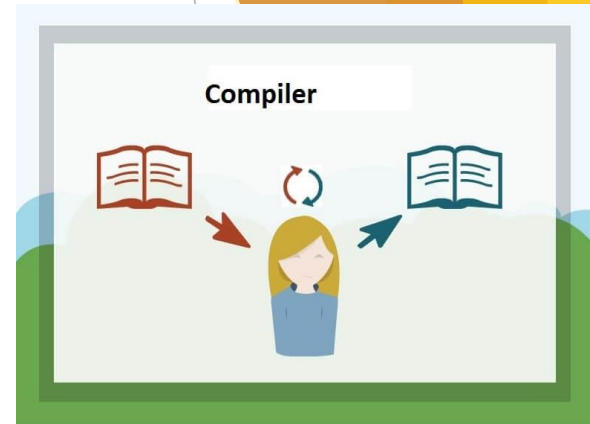
Programas - interpretadores

Nem todas as linguagens de programação de alto-nível são compiláveis, existem ainda as linguagens interpretáveis - tudo dependerá da linguagem de programação em uso!

Assim, além dos **compiladores**, existem ainda os **interpretadores**, que servem o mesmo princípio - leem o código que os humanos escrevem e fazem a tradução para código-máquina - para o computador também o conseguir entender.

As diferenças entre ambos está na forma como a “tradução” do código é feita:

- **No caso do compilador**, este recebe no código fonte na íntegra e gera um ficheiro já todo traduzido, que será mais tarde executado pelo computador - o compilador atua apenas uma única vez e o seu trabalho de tradução fica feito!
- **No caso do interpretador**, este funciona como um tradutor em tempo real: vai lendo aos poucos o código e traduzindo ao computador o que é para fazer - note-se que um interpretador nunca gera um ficheiro traduzido, a tradução vai ocorrendo somente ao longo da execução do programa.



Programas - diferenças entre linguagens

Abaixo podemos ver o antes e depois da compilação\interpretação e as diferenças entre ambas.

```
1 #include <iostream>
2 using namespace std;
3
4 int main() {
5     int numeroUser;
6     int numeroAdivinhar = 34;
7
8     cout << "Diga um número de 0 a 100: ";
9     cin >> numeroUser;
10
11     if(numeroUser == numeroAdivinhar)
12         cout << "Adivinhou!";
13     else
14         cout << "Errou!";
15 }
```

O que nós humanos escrevemos numa linguagem de programação de alto nível (no caso a linguagem C/C++)

```
01001101 01011010 00111111 00111111
00111111 00111111 00111111 00111111
00111111 00111111 00111111 00111111
00111111 00111111 00111111 00111111
00111111 00111111 00111111 00111111
00111111 00111111 00111111 00111111
01000000 00111111 00111111 00111111
00111111 00111111 00111111 00111111
00111111 00111111 00111111 00111111
00111111 00111111 00111111 00111111
00111111 00111111 00111111 00111111
00111111 00111111 00111111 00111111
00111111 00111111 00111111 00111111
00111111 00111111 00111111 00111111
00111111 00111111 00111111 00111111
00111111 00111111 00111111 00111111
00111111 00111111 00001001 00111111
00100001 00111111 00111111 (...)
```

O que o computador irá ler depois da linguagem de alto nível ser compilada \ interpretada numa linguagem de baixo nível.

Programas - gerações das linguagens

Ao longo dos anos, as linguagens de programação foram evoluindo, sempre tendo em vista o seu aperfeiçoamento, por forma a se tornarem mais eficientes e simples.

- **1º geração:** Linguagem máquina (**binário**) - linguagem nativa de qualquer microprocessador
- **2º geração:** Linguagem de montagem (**assembly**) - derivado da linguagem máquina, utiliza abreviaturas, mais fácil de entender pelo humano, ainda assim confuso e complexo!
- **3º geração:** Linguagem procedimental - aqui nasceram as linguagens de **programação de alto nível**, direcionadas para uma compreensão simples por parte do humano - C\C++, Java, C#, PHP, etc.
- **4º geração:** Linguagens com um grau mais alto de abstração - linguagens mais focadas na obtenção e formatação de dados - SQL, XML, HTML, etc.
- **5º geração:** Linguagens voltadas a Inteligência artificial - não se “programa”, apenas se indicam as restrições que existem para um determinado problema e o computador tenta (por via de várias tentativas e aprendizagem) obter uma solução viável.

Fases de desenvolvimento

Problema: Criar um bolo de chocolate novo para uma pastelaria

Entrada de dados:
Ingredientes a utilizar



Algoritmo:
Planeamento da receita



Saída de dados:
Elaboração do bolo
("Codificação" + "compilação")



i Uma **receita** nada mais é do que um conjunto de passos sequenciais (**algoritmo**), que relacionam a combinação dos ingredientes (**dados de entrada**) com vista à obtenção do produto final (**dados de saída**).



Refinamento e depuração

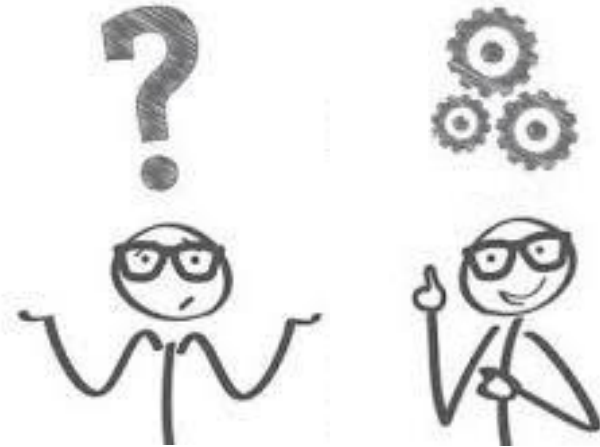


Fase de testes, deteção e correção de erros

Algoritmos

Algoritmos - definição

- Trata-se de um conjunto de instruções, com o intuito de descrever por passos sequenciais a resolução para um determinado problema.
- Note-se que um algoritmo não tem necessariamente uma relação direta com programas computacionais, a execução e interpretação de um algoritmo pode ser realizada tanto por um computador como até por um humano.



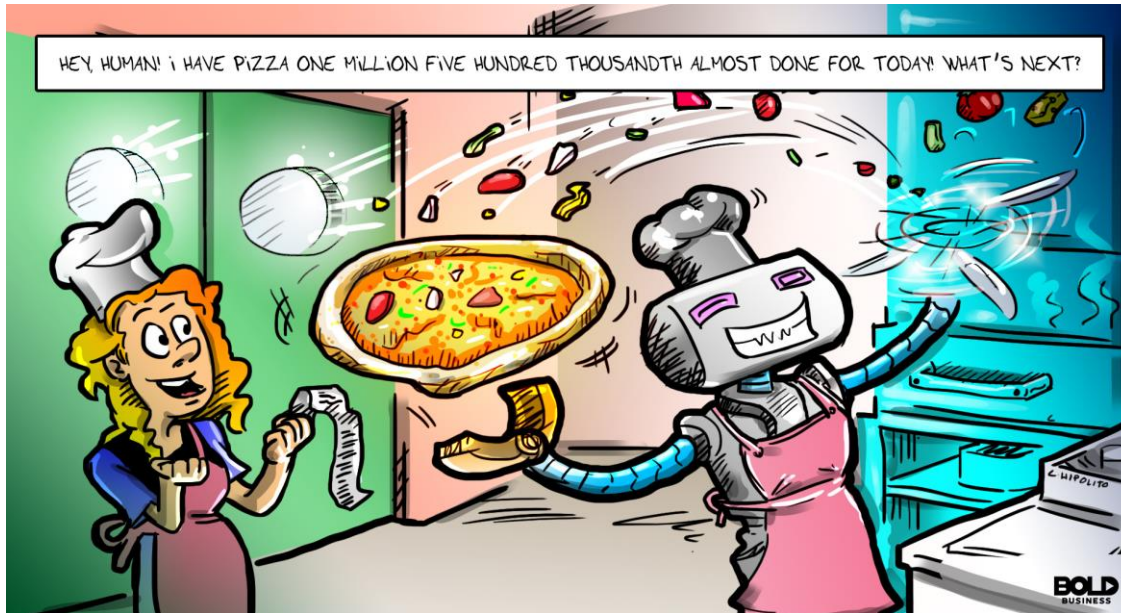
Algoritmos

- O nosso dia-a-dia já é composto de algoritmos.
- Lavar os dentes, fazer o balanço anual de contas, colocar loiça na máquina, mandar um email, cortar a relva, fazer café, e tantas outras tarefas, podem ser descritas num algoritmo - uma vez que todas se definem por uma sequência de passos, destinados à resolução de um determinado problema.



Algoritmo - para que servem

- Algoritmos, na computação, tratam-se de uma forma simples e generalizada de entendermos como “ensinar” devidamente uma máquina a realizar determinadas tarefas.

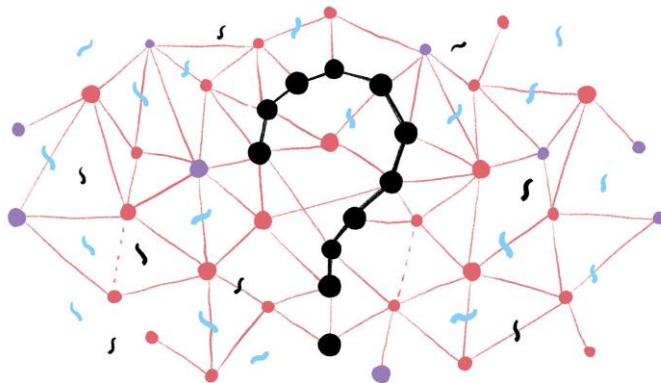


Formas de representação

Algoritmo - características

- Algoritmos podem ser representados na narrativa, fluxograma ou pseudocódigo - esboçando ao nível lógico, o funcionamento da nossa futura aplicação.
- Estes devem ter a indicação de um início e fim. Sendo que a sua interpretação, não deve dar margem para uma duplo entendimento.
- Note-se que **não existem regras estandardizadas** para representações algorítmicas, mas tenta-se seguir um modelo comum.

Um algoritmo em conjunto com uma linguagem de programação resultará num programa!



Algoritmo - formas de representação

- Analisemos a baixo as várias formas de representação de um algoritmo que recebe 3 números e apresenta o resultado da multiplicação destes:

Narrativa:

Algoritmo: Multiplicador de 3 números
Início
Pedir 3 números
Multiplicar os 3 números pedidos e guardar o resultado
Apresentar o resultado obtido
Fim

Fluxograma:



Pseudocódigo:

```
inteiro n1, n2, n3  
leia(n1)  
leia(n2)  
leia(n3)  
  
inteiro res = n1 * n2 * n3  
escreva (res)
```

Formas de representação

Narrativa

Algoritmo - Narrativa

- A narrativa focasse na descrição textual (humana e nativa) de qual o comportamento que o programa deverá seguir, por forma a solucionar o nosso problema.
- **Esta é a forma mais livre de representar algoritmos** - as únicas limitações serão sempre encontradas ao nível do hardware, uma vez que não podemos esperar que um computador, por exemplo: abra as janelas de casa, se ele não tiver nele ligados mecanismos (outro hardware) para essa tarefa ser realizada.

Exemplo de uma narrativa:

Algoritmo: Multiplicador de 3 números
Início
Pedir 3 números
Multiplicar os 3 números pedidos e guardar o resultado
Apresentar o resultado obtido
Fim



Narrativa:

Trata-se apenas de uma descrição livre, escrita na língua nativa.

Algoritmo - Narrativa

- Mas comecemos por representar algoritmos (na narrativa) com base no nosso dia-a-dia, lembrando que todo o algoritmo, deve ter um nome, um início, um fim e uma sequencia de instruções.

Estes seriam os passos sequenciais que normalmente realizamos para “tomar banho”

Algoritmo tomar banho
Início
Tirar a roupa
Abrir a torneira
Ensaboar-se
Enxaguar o corpo
Passar shampoo no cabelo
Enxaguar o cabelo
Fechar a torneira
Fim

Nome do algoritmo

Corpo do algoritmo

Conjunto de instruções sequencias

Algoritmo - Narrativa

Exercício 2

Desenvolva um algoritmo (com o máximo de 10 passos) de como **ver TV**.

Tenha em consideração que:

- A TV e a Box estão inicialmente desligados
- Você quer ver TV sentado no sofá
- Você quer apenas ver 20 minutos de TV
- Depois de ver TV não deverá deixar os aparelhos ligados.

► Cácula:

Algoritmo tomar banho

Início

Tirar a roupa

Abrir a torneira

Ensaboar-se

Enxaguar o corpo

Passar shampoo no cabelo

Enxaguar o cabelo

Fechar a torneira

Fim

Algoritmo - Narrativa

Exercício 2 (Solução)

Algoritmo ver TV

Início

Ligar a TV

Ligar a Box

Sentar-me no sofá

Ver TV 20 minutos

Levantar-me do sofá

Desligar a TV

Desligar a Box

Fim

Algoritmo - Narrativa

- Note-se que ao formular um algoritmo, a ordem da sequência de instruções é importante! Imagine que está a ensinar alguém a ver TV, terá que lhe indicar os passos certos para que a pessoa faça devidamente esta tarefa!

Algoritmo ver TV
Início
Sentar-me no sofá
Desligar a TV
Ver TV 20 minutos
Ligar a TV
Ligar a Box
Levantar-me do sofá
Desligar a Box
Fim

Se trocarmos a ordem das instruções, faz sentido ver-mos TV desta forma?

Algoritmo - Narrativa

Exercício 3

Desenvolva um algoritmo (com o máximo de 10 passos) de como **beber sumo**.

Tenha em consideração que:

- O sumo está no frigorífico (que está inicialmente fechado)
- O sumo está numa garrafa (fechada)
- Irá beber de um copo
- No final a garrafa e o frigorífico necessitam de estar fechados

Nota: Imagine que está a escrever num papel os passos para ensinar alguém a realizar esta tarefa.

► Cábula:

Algoritmo tomar banho
Início
Tirar a roupa
Abrir a torneira
Ensaboar-se
Enxaguar o corpo
Passar shampoo no cabelo
Enxaguar o cabelo
Fechar a torneira
Fim

Algoritmo - Narrativa

Exercício 3 (Solução)

Algoritmo Beber sumo

Início

Buscar um copo

Abrir o frigorifico

Tirar embalagem

Abrir embalagem

Colocar sumo no copo

Beber sumo

Fechar a embalagem

Fechar o frigorifico

Arrumar o copo para lavar

Fim

Algoritmo - Narrativa

Exercício 4

Desenvolva um algoritmo (com o máximo de 10 passos) para **receber o pagamento** de um cliente.

Tenha em consideração que:

- O cliente pode precisar de troco
- O cliente pode quer fatura
- O cliente pode precisar de um saco

Nota: Imagine que está a escrever num papel os passos para ensinar alguém a realizar esta tarefa.

► Cábula:

Algoritmo tomar banho

Início

Tirar a roupa

Abrir a torneira

Ensaboar-se

Enxaguar o corpo

Passar shampoo no cabelo

Enxaguar o cabelo

Fechar a torneira

Fim

Algoritmo - Narrativa

Exercício 4 (Solução)

Algoritmo Receber pagamento

Início

Informar o cliente que são 10€

Receber o dinheiro do cliente

Pedir contribuinte ao cliente

Registrar fatura no sistema

Retornar 10€ de troco ao cliente

Entregar a fatura ao cliente

Colocar o produto dentro de um saco

Entregar o produto ao cliente

Fim

Algoritmo - Narrativa

Exercício 5

Desenvolva um algoritmo representado em forma de narrativa que mande **somar dois números** e no fim apresente o seu resultado.

Tenha em consideração que:

- Necessita de solicitar os números, efetuar o cálculo e apresentar os resultados

Nota: Não se esqueça que terá que inicialmente solicitar os dois números, realizar o calculo e apresentar o resultado no fim.

► Cábula:

Algoritmo: Multiplicador de 3 números

Início

Pedir 3 números

Multiplicar os 3 números pedidos e
guardar o resultado

Apresentar o resultado obtido

Fim

Algoritmo - Narrativa

Exercício 5 (solução)

Algoritmo: somar 2 números

Início

Pedir 2 números

Somar os 2 números pedidos

guardar o resultado

Apresentar o resultado obtido

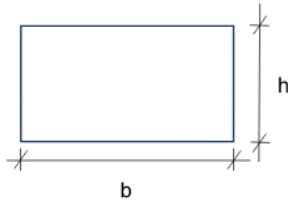
Fim

Algoritmo - Narrativa

Exercício 6

Desenvolva um algoritmo representado em forma de narrativa que permita calcular a **área de um retângulo**.

Fórmula :



$$\text{Área} = \text{Base} * \text{Altura}$$

Nota: Não se esqueça que terá que solicitar a altura e a base, para seguidamente conseguir efetuar o cálculo e apresentar o resultado.

► Cácula:

Algoritmo: Multiplicador de 3 números

Início

Pedir 3 números

Multiplicar os 3 números pedidos e guardar o resultado

Apresentar o resultado obtido

Fim

Algoritmo - Narrativa

Exercício 6 (solução)

Algoritmo: área de um retângulo

Início

Pedir a base

Pedir a altura

Multiplicar a base pela altura e
guardar o resultado

Apresentar o resultado obtido

Fim

OU

Algoritmo: área de um retângulo

Início

Pedir a base e da altura

Multiplicar a base pela altura e
guardar como valor da área

Apresentar o valor da área

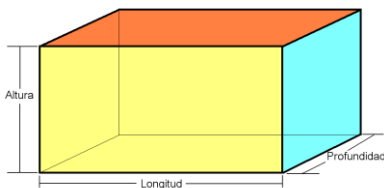
Fim

Algoritmo - Narrativa

Exercício 7

Desenvolva um algoritmo representado em forma de narrativa que permita calcular o **volume de um ortoedro**.

Fórmula:



$$\text{Volume} = \text{altura} * \text{largura} * \text{profundidade}$$

Nota: Não se esqueça que terá que solicitar a altura, a largura e a profundidade, para seguidamente efetuar o cálculo, e apresentar o resultado.

► Cábula:

Algoritmo: Multiplicador de 3 números

Início

Pedir 3 números

Multiplicar os 3 números pedidos e guardar o resultado

Apresentar o resultado obtido

Fim

Algoritmo - Narrativa

Exercício 7 (solução)

Algoritmo: volume de um ortoedro

Início

Pedir o tamanho a altura, a largura e a profundidade

Multiplicar os 3 valores e guardar como valor do volume

Apresentar o valor do volume

Fim

Algoritmo - Quebra-Cabeças 2

Uma vez um agricultor foi ao mercado e comprou um lobo, uma ovelha e uma alface.

No caminho para casa, como havia chovido muito, o rio encheu até à margem, tendo o agricultor que fazer a travessia a nado, podendo apenas **levar uma das suas compras por cada viagem** - o lobo, a ovelha ou a alface.

Saiba-se que se forem deixados sozinhos sem a presença do agricultor, o lobo comerá a ovelha e a ovelha comerá a alface.

O desafio do agricultor é passar as suas compras para a margem oposta do rio, deixando-as intactas.



Quebra-Cabeças 2 (Solução 1 / 7)



Quebra-Cabeças 2 (Solução 1 / 7)



Quebra-Cabeças 2 (Solução 1 / 7)



Quebra-Cabeças 2 (Solução 1 / 7)



Quebra-Cabeças 2 (Solução 1 / 7)



Quebra-Cabeças 2 (Solução 2 / 7)



Quebra-Cabeças 2 (Solução 2 / 7)



Quebra-Cabeças 2 (Solução 3 / 7)



Quebra-Cabeças 2 (Solução 3 / 7)



Quebra-Cabeças 2 (Solução 3 / 7)



Quebra-Cabeças 2 (Solução 4 / 7)



Quebra-Cabeças 2 (Solução 4 / 7)



Quebra-Cabeças 2 (Solução 4 / 7)



Quebra-Cabeças 2 (Solução 5 / 7)



Quebra-Cabeças 2 (Solução 5 / 7)



Quebra-Cabeças 2 (Solução 5 / 7)



Quebra-Cabeças 2 (Solução 6 / 7)



Quebra-Cabeças 2 (Solução 6 / 7)



Quebra-Cabeças 2 (Solução 6 / 7)



Quebra-Cabeças 2 (Solução 7 / 7)



Quebra-Cabeças 2 (Solução 7 / 7)



Quebra-Cabeças 2 (Solução 7 / 7)



Quebra-Cabeças 2 (Solução 7 / 7)



Quebra-Cabeças 2 - algoritmo

Algoritmo para a resolução do Quebra-Cabeças:

Algoritmo atravessar o rio

Início

Levar a ovelha

Voltar

Levar o lobo

Deixar o lobo e trazer a ovelha de volta

Deixar a ovelha e levar a alface

Deixar a alface e voltar sem nada

Levar a ovelha

Fim

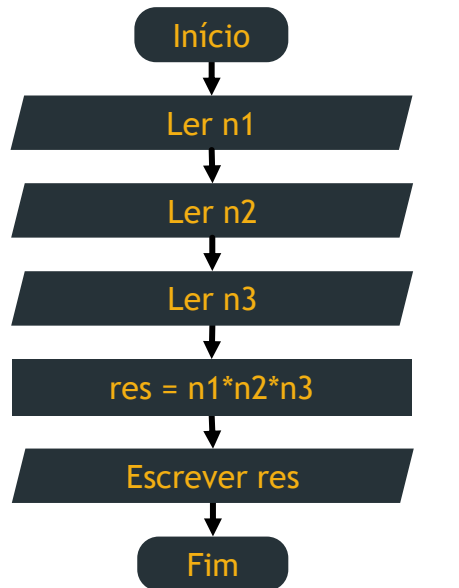
Formas de representação

Fluxograma

Algoritmos - Fluxograma

- O fluxograma, é um tipo de diagrama que procura esquematizar de forma ilustrativa, o fluxo de dados (entrada, processamento e saída) de um qualquer algoritmo.

Exemplo de um fluxograma:



i



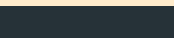



Fluxograma:

Descreve visualmente os passos que um programa deverá tomar.

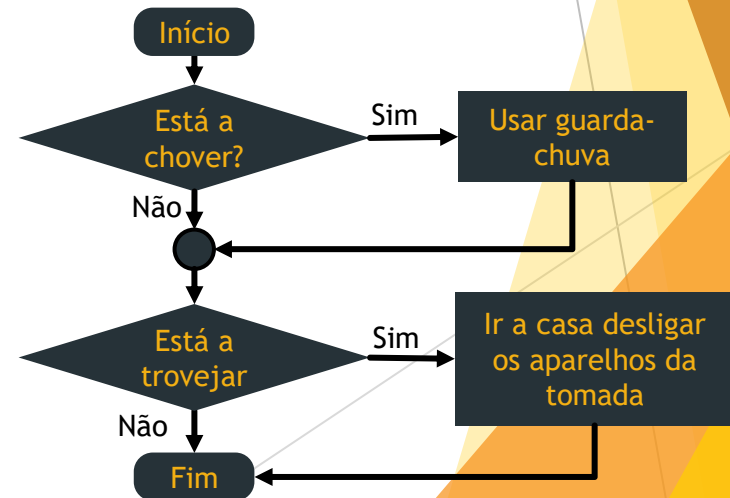
Algoritmos - Fluxograma

- Com um fluxograma podemos representar, a partir de símbolos, o início e fim do algoritmo, assim como o fluxo (de processamento) da informação, o processamento, a tomada de decisões e a entrada e saída de dados.
- Nota:** Os termos usados dentro dos símbolos, devem simplificados ao máximo!

Símbolos usados num fluxograma:

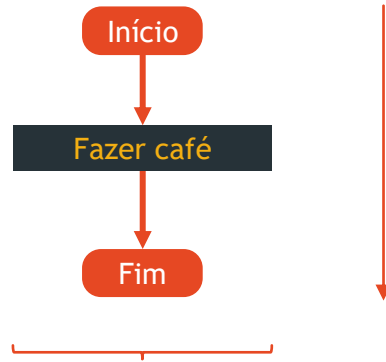
Símbolos	Descrição
	Indicação do início ou fim do algoritmo
	Identifica o fluxo de processamento da informação
	Processamento
	Tomada de decisões
	Entrada ou saída de dados
	Conetor

Exemplo de um algoritmo representado em fluxograma:



Algoritmos - Fluxograma

- A indicação de **início** e **fim** e as setas de fluxo, no fluxograma, servem somente para entendermos qual o sentido que o nosso algoritmo deve tomar assim como qual o seu início e fim!



Sabendo o início e o fim, e seguindo o sentido do fluxo, encontramos facilmente o ponto de partida, assim como todos os passos que foram tomados até encontrarmos a conclusão!

i

Este algoritmo remete para que inicialmente se faça café, dando seguimento à conclusão imediata de execuções.

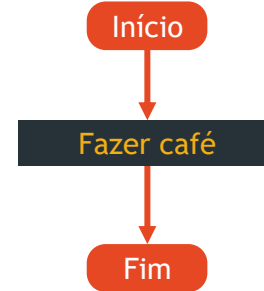
Algoritmos - Fluxograma

Exercício 8

Desenvolva um fluxograma que mande somente fechar a janela.

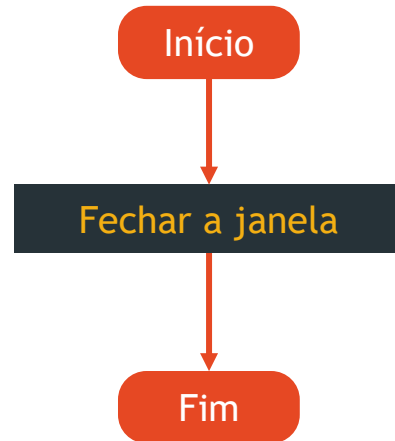
Nota: Não se esqueça que o fluxograma deverá indicar um início e um fim, assim como o fluxo que deverá tomar!

► Cábula:



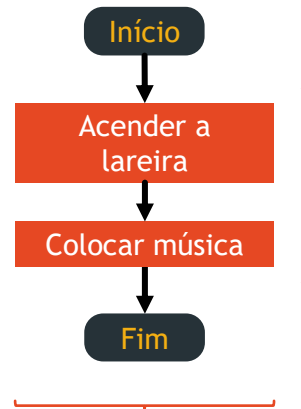
Algoritmos - Fluxograma

Exercício 8 (solução)



Algoritmos - Fluxograma

- A indicação de **processamento** do fluxograma, serve para indicarmos que uma determinada ação deve ser disputada naquele momento.



Cada vez que precisamos de mandar realizar ações ao sistema, devemos usar estes símbolos.

Nota: cada ação diferente deve ser representada separadamente e entre cada uma deve existir uma seta a indicar o fluxo.



Este algoritmo remete para que inicialmente se acenda a lareira, seguidamente se coloque musica, seguindo para a conclusão imediata de execuções.

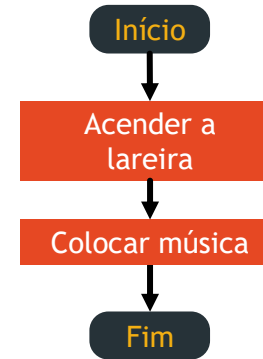
Algoritmos - Fluxograma

Exercício 9

Desenvolva um fluxograma que mande somente abrir os estores, seguido de apagar as luzes.

Nota: Não se esqueça que o fluxograma deverá indicar um início e um fim, assim como o fluxo que deverá tomar!

► Cábula:



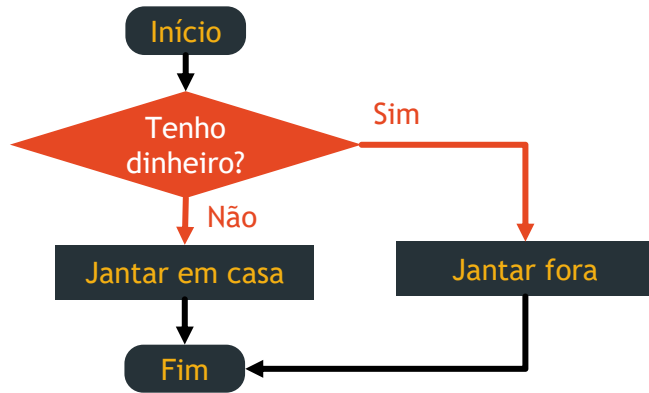
Algoritmos - Fluxograma

Exercício 9 (solução)



Algoritmos - Fluxograma

- A indicação de **tomada de decisões** do fluxograma, serve para indicarmos que perante uma determinada situação ser verdadeira ou falsa, podemos tomar dois rumos.
- **Note-se que uma tomada de decisão deve ser formulada como pergunta!**



Sempre que precisemos de validar alguma decisão, devemos utilizar este símbolo.

Note-se que em cada ramo existe um “sim” e um “não” que deverão estar sempre presentes.

Os pontos de saída das setas de fluxo (de qualquer elemento do fluxograma) são arbitrários!

i

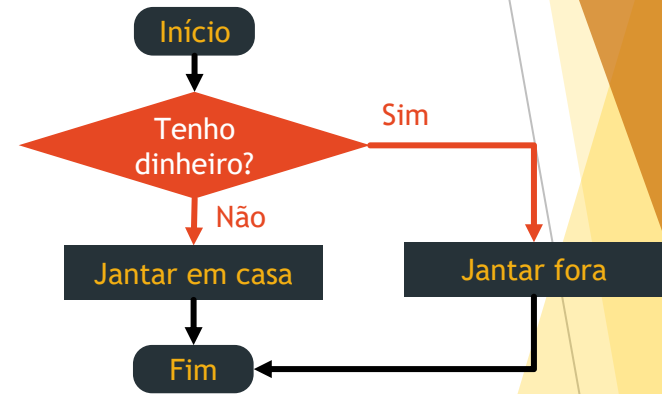
Este algoritmo remete para que inicialmente se verifique se “tenho dinheiro”, caso afirmativo: vou jantar fora e conclui-se imediatamente a execução. caso negativo: janto em casa e conclui-se imediatamente a execução.

Algoritmos - Fluxograma

Exercício 10

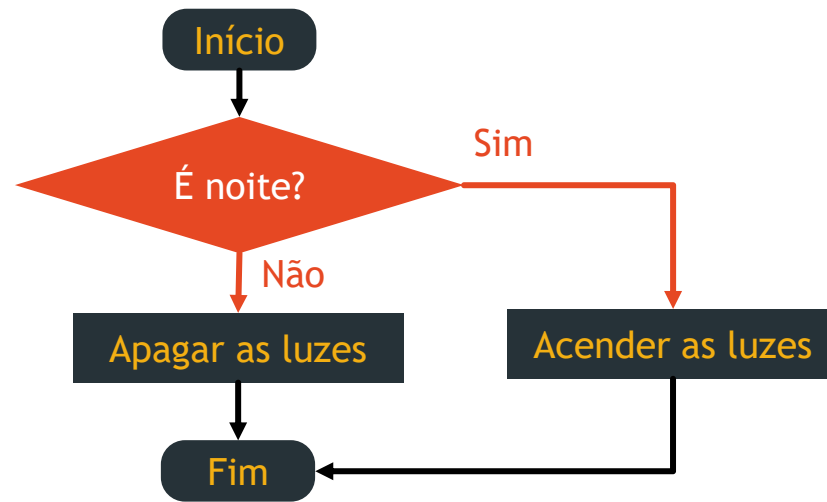
Desenvolva um fluxograma que verifique se caso seja noite, para mandar ligar as luzes, caso não seja, para as apagar.

► Cácula:



Algoritmos - Fluxograma

Exercício 10 (solução)



Algoritmos - Fluxograma

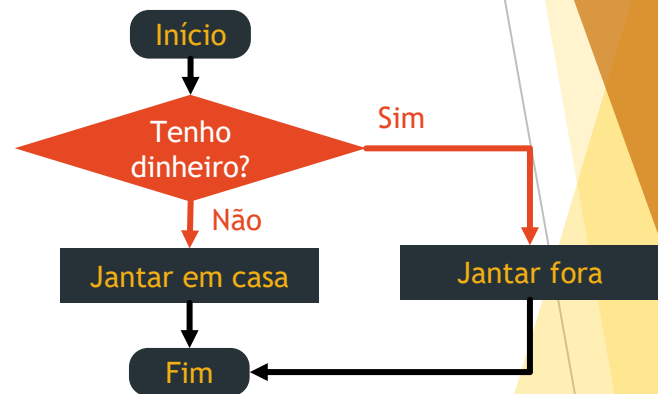
Exercício 11

Desenvolva um fluxograma que comece por lavar o carro, seguidamente o seque e no fim verifica se está com pouco brilho, caso esteja, deverá ser aplicada cera, caso não esteja, não deverá ser aplicada cera. Após isto, o carro deverá ser arrumado na garagem e a execução do algoritmo terminada.

Nota: Comece por distinguir individualmente o que é processamento e o que são tomadas de decisões, seguidamente faça ligar devidamente o fluxo entre os vários elementos.

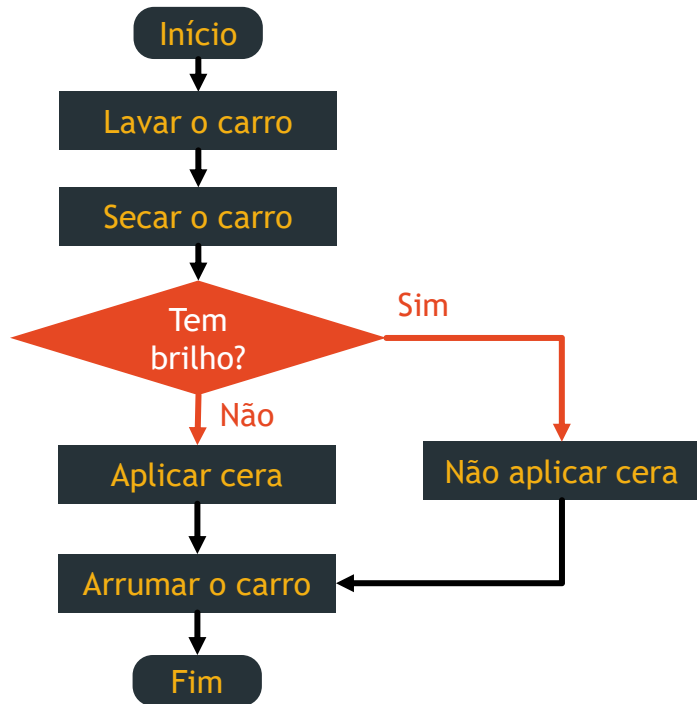
Não se esqueça que na tomada de decisões devemos sempre indicar um ramo para o “sim” e outro para o “não” e estes devem estar mencionados nas saídas de fluxo.

► Cábula:

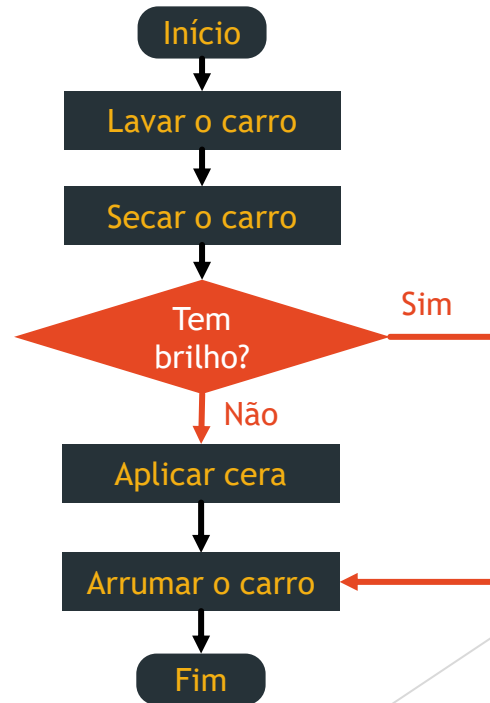


Algoritmos - Fluxograma

Exercício 11 (solução)



OU:



Caso o carro já tenha brilho, se avançarmos logo para o arrumar (na garagem), o efeito será o mesmo, uma vez que em termos lógicos, também não aplicamos cera!

Algoritmos - Fluxograma

Exercício 12

Desenvolva um fluxograma para trocar uma lâmpada.

Comece por desligar o interruptor da lâmpada, seguidamente verifique se existe lâmpada no candeeiro:

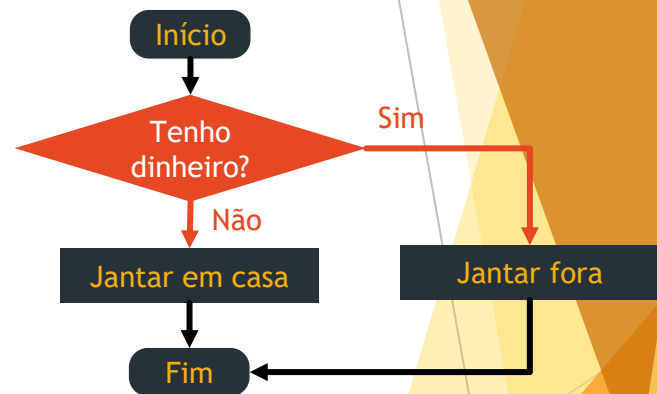
- se existir: verifique se o filamento está rompido:
 - Caso esteja: troque a lâmpada.
 - Caso não esteja: reponha a lâmpada.
- Se não existir: coloque uma lâmpada nova

Por último deverá acender o interruptor

Nota: Comece por distinguir individualmente o que é processamento e o que são tomadas de decisões, seguidamente faça ligar devidamente o fluxo entre os vários elementos.

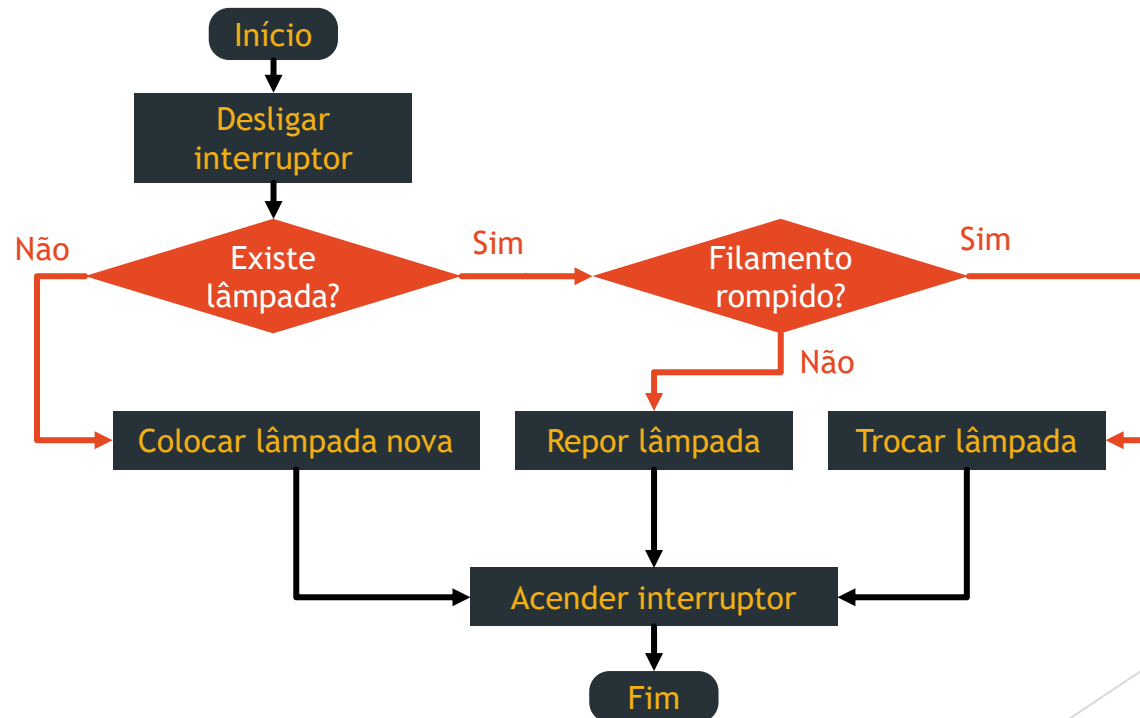
Lembre-se que agora tem 2 decisões a tomar, e não existe nenhuma regra que nos previna de ligar várias em cadeia!

► Cábula:



Algoritmos - Fluxograma

Exercício 12 (solução)

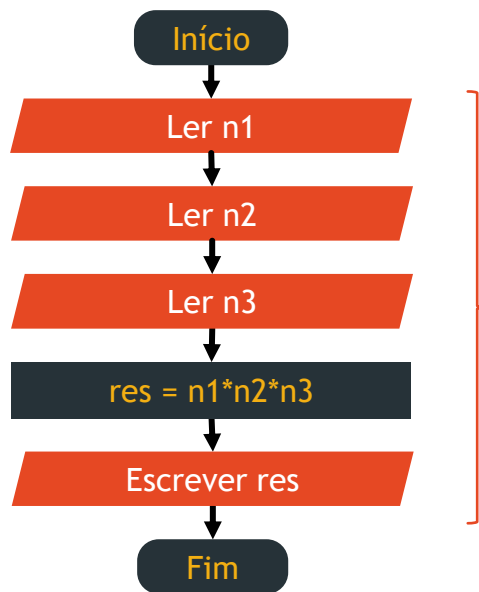


Algoritmos - Fluxograma

- A **entrada ou saída de dados** do fluxograma, serve para solicitar ou apresentar dados ao utilizador.
- A entrada de dados (solicitação) é por norma realizada pedindo ao utilizador para introduzir o valor em conveniência. Já a saída de dados, serve para apresentar um valor, que já foi anteriormente solicitado ou até processado (calculado).



Este algoritmo remete para que inicialmente sejam (separadamente) solicitados 3 valores (guardados em n1, n2, e n3), seguidamente seja feita a multiplicação destes e guardado o resultado em “res”, por fim, é apresentado o resultado (escrevendo o que está em “res”). Logo após, conclui-se imediatamente a execução.

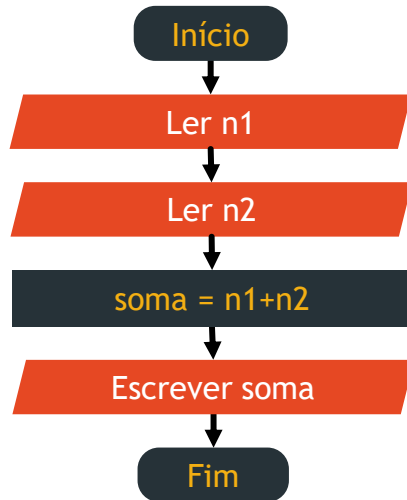


Sempre que exista a necessidade de pedir ou apresentar um valor, devemos usar o símbolo em questão.

Note-se que para processarmos um calculo, devemos usar o símbolo de processamento (retângulo)

Algoritmos - Fluxograma

- Na **entrada ou saída de dados** do fluxograma, devemos considerar o uso de variáveis, estas receberão o valor do utilizador e apresentarão o valor nestas guardadas.



i n1, n2 e soma tratam-se de variáveis.
“ler” representa a entrada de dados, sendo que “escrever” representa a saída.

i

Ler	n1
Escrever	soma

Comando Variável

i

soma = n1+n2

Caso queiramos fazer um calculo, utilizamos uma variável juntamente com o sinal de atribuição “=”, seguido do calculo a efetuar. Isto fará com que a variável “soma” receba o valor de n1 + n2.

Algoritmos - Fluxograma

Exercício 13

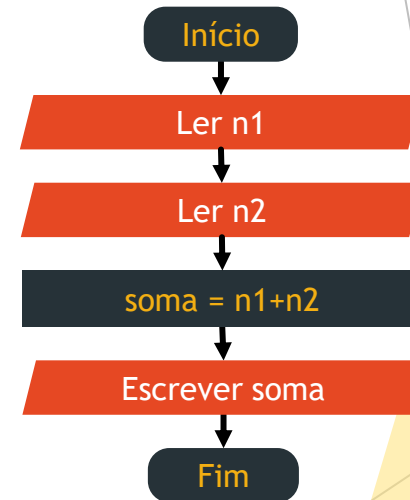
Desenvolva um fluxograma para calcular a área de um retângulo.

Fórmula:

$$\text{Área} = \text{Base} * \text{Altura}$$

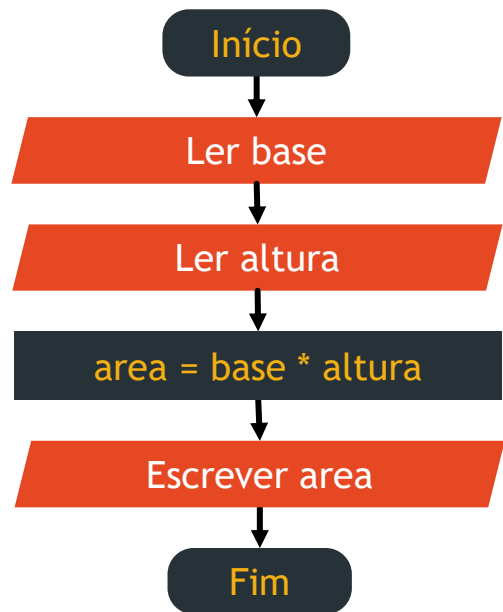
Nota: Comece por solicitar (entrada de dados) a base e depois a altura, seguidamente apresente o valor (saída de dados) da área atendendo ao cálculo da fórmula acima.

► Cábula:



Algoritmos - Fluxograma

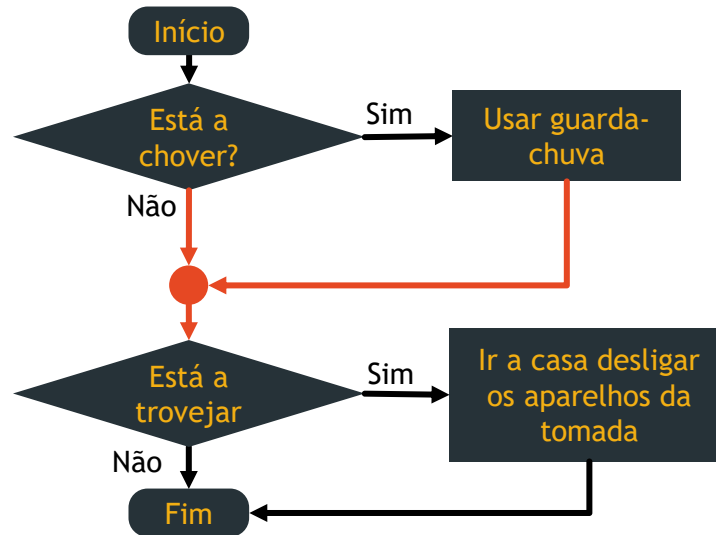
Exercício 13 (solução)



Não se trata de uma regra nos fluxogramas, mas na programação, as variáveis não devem ter acentos ou “ç”! Como é o caso de “área”.

Algoritmos - Fluxograma

- O **conetor** do fluxograma, serve somente para unir vários pontos de fluxo - apenas o devemos usar quando os elementos do fluxograma não nos permitem devidamente indicar o fluxo de dados.



Algoritmos - Fluxograma

Exercício 14

Desenvolva um fluxograma que verifique se esta a fazer sol:

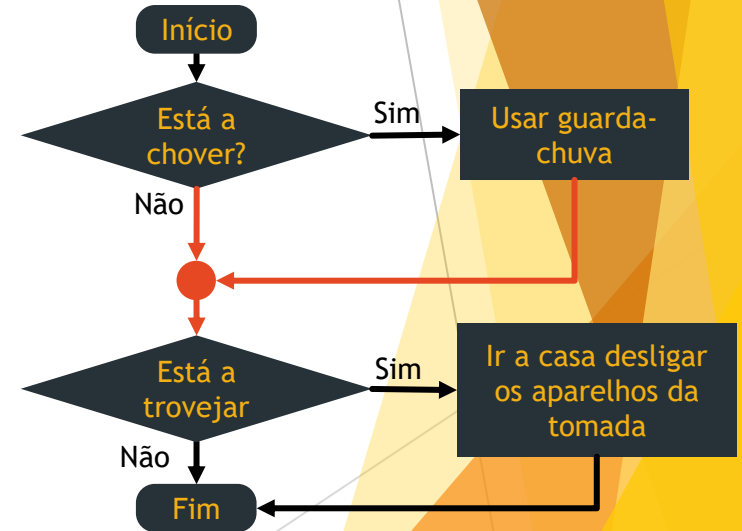
- Se estiver: abrir os estores
- Se não estiver: fechar os estores

Seguidamente (para ambos os casos) verifique se são 8 horas:

- Caso sejam 8 horas: colar música ambiente, ligar a TV e terminar a execução de tarefas.
- Caso não sejam: simplesmente terminar a execução de tarefas.

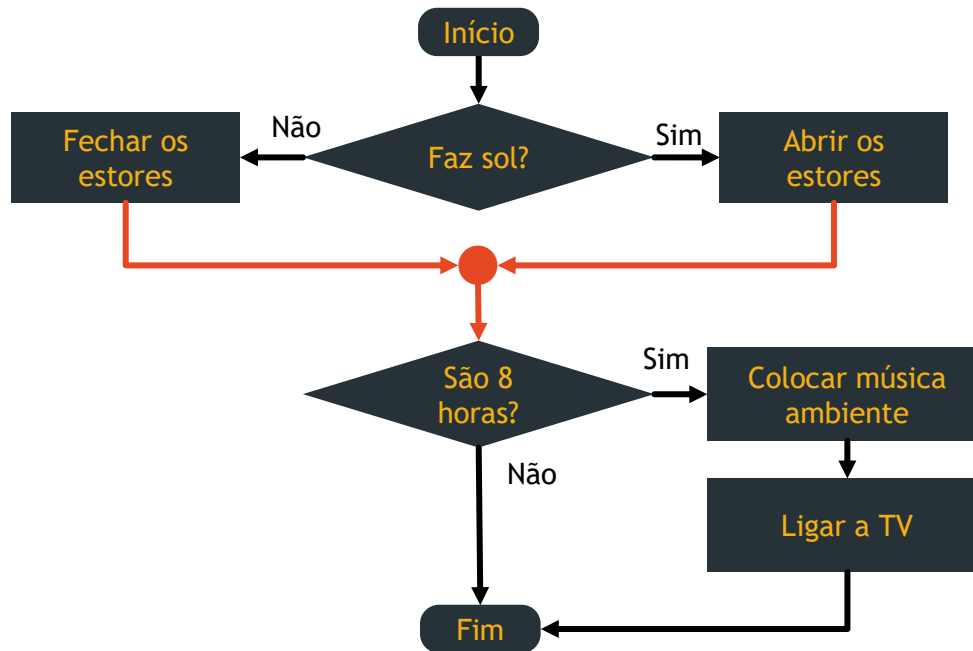
Nota: Utilize o símbolo de conector para desenvolver este fluxograma.

► Cábula:



Algoritmos - Fluxograma

Exercício 14 (solução)



Formas de representação

Pseudocódigo

Algoritmos - Pseudocódigo

- Trata-se de uma representação genérica e aproximada do código-fonte, utilizando uma linguagem simples e direta, representada por instruções.
- Nesta representação algorítmica, também não existem regras, no entanto, para nos aproximarmos da programação, iremos defini-las, para que nos preparemos para o desenvolvimento futuro.

Exemplo de um pseudocódigo:
(Escrito em pseudocódigo, em “Portogol”)

```
programa apresentaNumero
{
    funcao inicio()
    {
        inteiro num;
        escreva("Indique um número: ")
        leia (num)
        escreva ("O número que indicou é: " + num)
    }
}
```

Exemplo do mesmo código mas em código-fonte:
(Escrito na linguagem Java)

```
public class apresentaNumero {
    public static Scanner scan = new Scanner(System.in);

    public static void main(String[] args) {
        int num;
        System.out.println("Indique um número: ");
        num = scan.nextInt();
        System.out.println("O número que indicou é: " + num);
    }
}
```

Pseudocódigo - Variáveis e dados

- ▶ O que é uma variável?

Imaginar como sendo uma caixa vazia



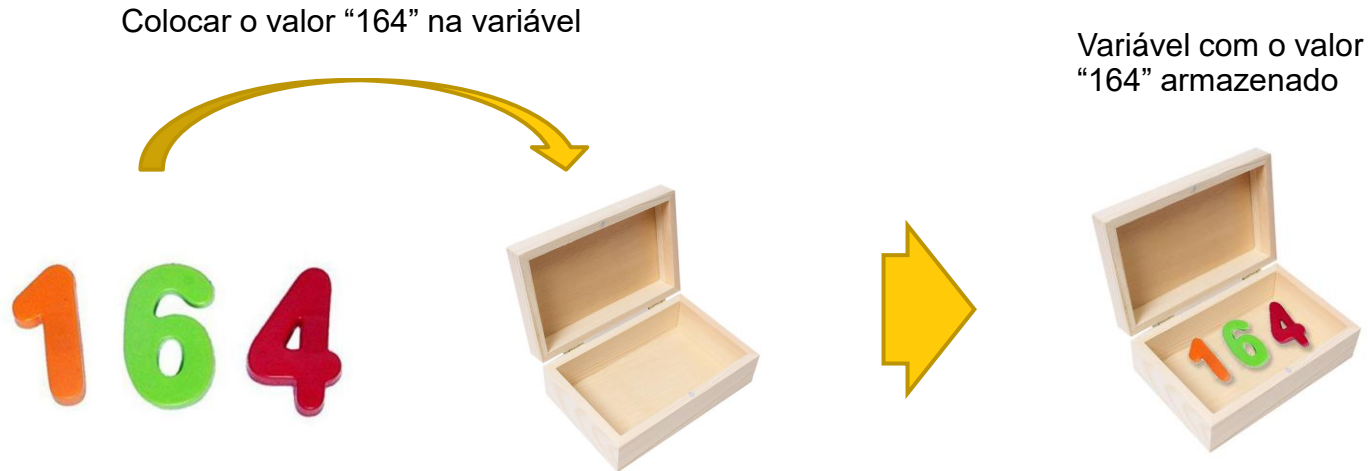
- ▶ O que são dados?

Imaginar como caracteres magnéticos



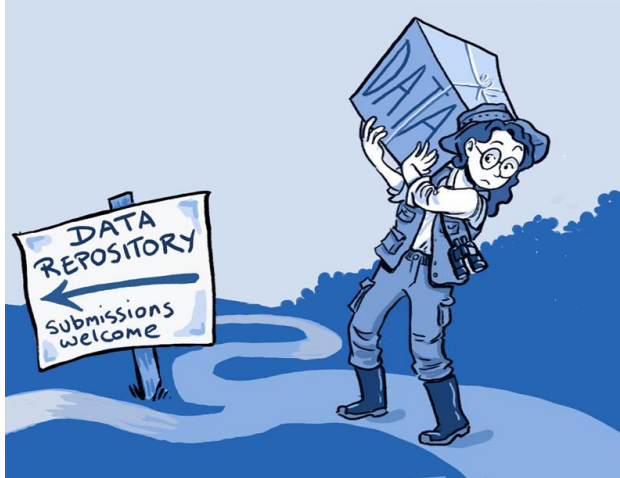
Pseudocódigo - Variáveis e dados

- Como se armazenam dados em variáveis



Pseudocódigo - Tipos de dados

- Existem vários tipos de dados, cada um destinado a representar tipos de informação diferente.
- Os tipos de dados auxiliam as variáveis, dizendo-lhes com que tipo de informação devem estar preparadas para trabalhar.



Pseudocódigo - Tipos de dados

- Os tipos de dados podem-se subdividir em 3 categorias:
 - Numéricos - **inteiros ou decimais**
 - Estados
 - Lógico – **Verdadeiro** ou **Falso** - **dois estados possíveis**
 - Caracter – 1 caractere: número, letra, símbolo – **256 estados possíveis**
 - Texto - **texto livre**

	Tipo	Descrição
Numérico	inteiro	Define variáveis numéricas, destinadas a armazenar números do tipo inteiro (sem casas decimais).
	real	Define variáveis numéricas, destinadas a armazenar números inteiros e números do tipo decimal (com casas decimais).
Estados	logico	Define variáveis destinadas a guardar valores lógicos (verdadeiro ou falso)
	caracter	Define variáveis destinadas a armazenar apenas um caractere
Texto	cadeia	Define variáveis destinadas a armazenar texto (considerado também como sendo uma cadeia de carateres ou strings)

Pseudocódigo - Tabela ASCII

- American Standard Code for Information Interchange (Código Padrão Americano de Intercâmbio de Informações)
 - A tabela ASCII trata-se uma tabela indexada de caracteres
 - Cada caractere é representado por uma posição numérica
 - Os computadores somente interpretam números

The screenshot shows a debugger interface with several panels. The top-left panel displays CPU registers: Acc (15), Xreg (80), SP (00FF), PC (CCR), and 0175 (111.I...). The top-right panel shows source code for 'project.asm' with comments and assembly instructions like 'lda #50', 'jsr delay', 'lda prta', and 'and #11100000q'. The middle-left panel lists variables: PORTA (\$00), PORTB (\$00), and PORT (\$00). The middle-right panel shows a table of breakpoints with columns for Address, Break After, Counter, BreakA, BreakX, and BreakSP. The bottom-left panel shows a command history with entries like 'PC loaded with Reset Vector.', '>NOCNT 021B', and '>BR 021B'. The bottom-right panel shows a debug window with 'F10:Debug'.

Address	Break After	Counter	BreakA	BreakX	BreakSP
START_VOICE	\$0001	no	\$44	\$55	\$FE
TEST7	-	YES	-	-	-
0344	\$0007	YES	-	-	-
CHECK_BUTTON	\$0001	no	-	-	-
available	-	no	-	-	-

i Ainda há não muito tempo, a tabela ASCII dava-nos suporte para “desenhar” interfaces mais apelativas do que um simples ecrã preto.

Pseudocódigo - Tabela ASCII

Dec	Hx	Oct	Char	Dec	Hx	Oct	Html	Chr	Dec	Hx	Oct	Html	Chr	Dec	Hx	Oct	Html	Chr
0	0	000	NUL (null)	32	20	040	 	Space	64	40	100	@	@	96	60	140	`	`
1	1	001	SOH (start of heading)	33	21	041	!	!	65	41	101	A	A	97	61	141	a	a
2	2	002	STX (start of text)	34	22	042	"	"	66	42	102	B	B	98	62	142	b	b
3	3	003	ETX (end of text)	35	23	043	#	#	67	43	103	C	C	99	63	143	c	c
4	4	004	EOT (end of transmission)	36	24	044	$	\$	68	44	104	D	D	100	64	144	d	d
5	5	005	ENQ (enquiry)	37	25	045	%	%	69	45	105	E	E	101	65	145	e	e
6	6	006	ACK (acknowledge)	38	26	046	&	&	70	46	106	F	F	102	66	146	f	f
7	7	007	BEL (bell)	39	27	047	'	'	71	47	107	G	G	103	67	147	g	g
8	8	010	BS (backspace)	40	28	050	((72	48	110	H	H	104	68	150	h	h
9	9	011	TAB (horizontal tab)	41	29	051))	73	49	111	I	I	105	69	151	i	i
10	A	012	LF (NL line feed, new line)	42	2A	052	*	*	74	4A	112	J	J	106	6A	152	j	j
11	B	013	VT (vertical tab)	43	2B	053	+	+	75	4B	113	K	K	107	6B	153	k	k
12	C	014	FF (NP form feed, new page)	44	2C	054	,	,	76	4C	114	L	L	108	6C	154	l	l
13	D	015	CR (carriage return)	45	2D	055	-	-	77	4D	115	M	M	109	6D	155	m	m
14	E	016	SO (shift out)	46	2E	056	.	.	78	4E	116	N	N	110	6E	156	n	n
15	F	017	SI (shift in)	47	2F	057	/	/	79	4F	117	O	O	111	6F	157	o	o
16	10	020	DLE (data link escape)	48	30	060	0	0	80	50	120	P	P	112	70	160	p	p
17	11	021	DC1 (device control 1)	49	31	061	1	1	81	51	121	Q	Q	113	71	161	q	q
18	12	022	DC2 (device control 2)	50	32	062	2	2	82	52	122	R	R	114	72	162	r	r
19	13	023	DC3 (device control 3)	51	33	063	3	3	83	53	123	S	S	115	73	163	s	s
20	14	024	DC4 (device control 4)	52	34	064	4	4	84	54	124	T	T	116	74	164	t	t
21	15	025	NAK (negative acknowledge)	53	35	065	5	5	85	55	125	U	U	117	75	165	u	u
22	16	026	SYN (synchronous idle)	54	36	066	6	6	86	56	126	V	V	118	76	166	v	v
23	17	027	ETB (end of trans. block)	55	37	067	7	7	87	57	127	W	W	119	77	167	w	w
24	18	030	CAN (cancel)	56	38	070	8	8	88	58	130	X	X	120	78	170	x	x
25	19	031	EM (end of medium)	57	39	071	9	9	89	59	131	Y	Y	121	79	171	y	y
26	1A	032	SUB (substitute)	58	3A	072	:	:	90	5A	132	Z	Z	122	7A	172	z	z
27	1B	033	ESC (escape)	59	3B	073	;	;	91	5B	133	[[123	7B	173	{	{
28	1C	034	FS (file separator)	60	3C	074	<	<	92	5C	134	\	\	124	7C	174	|	
29	1D	035	GS (group separator)	61	3D	075	=	=	93	5D	135]]	125	7D	175	}	}
30	1E	036	RS (record separator)	62	3E	076	>	>	94	5E	136	^	^	126	7E	176	~	~
31	1F	037	US (unit separator)	63	3F	077	?	?	95	5F	137	_	_	127	7F	177		DEL

Source: www.asciitable.com

128	Ç	144	É	160	á	176	☒	192	Ł	208	Ł	224	α	240	≡
129	ü	145	Æ	161	í	177	☒	193	ł	209	ŧ	225	β	241	±
130	é	146	æ	162	ó	178	☒	194	ṭ	210	ṭ	226	Γ	242	≥
131	â	147	ô	163	ú	179		195	ṭ	211	ṭ	227	π	243	≤
132	ä	148	ö	164	ñ	180	†	196	—	212	Ł	228	Σ	244	ƒ
133	à	149	ò	165	Ñ	181	†	197	†	213	ƒ	229	σ	245	Ƶ
134	â	150	û	166	°	182	†	198	†	214	ƒ	230	μ	246	+
135	ç	151	ü	167	°	183	†	199	†	215	†	231	τ	247	°
136	è	152	ý	168	¿	184	†	200	Ł	216	†	232	Φ	248	°
137	é	153	Ö	169	ƒ	185	†	201	ƒ	217	Ƶ	233	Θ	249	.
138	ê	154	Û	170	ƒ	186	†	202	Ł	218	ƒ	234	Ω	250	.
139	í	155	°	171	¼	187	†	203	†	219	■	235	δ	251	√
140	î	156	°	172	¼	188	†	204	†	220	■	236	∞	252	∞
141	ï	157	¾	173	½	189	†	205	=	221	†	237	φ	253	z
142	Ä	158	£	174	«	190	†	206	†	222	†	238	e	254	■
143	Å	159	ƒ	175	»	191	†	207	Ł	223	■	239	∩	255	

Source: www.LookupTables.com

Pseudocódigo - Tipos de dados

- Como identificar o tipo de dados necessário
 - É um numero? Tem casas decimais?
 - É um estado?
 - Tem 2 estados? Posso identificar como Verdade (sim) ou Falso (não)?
 - Tem mais do que 2 estados possíveis?
 - É textual?

Pseudocódigo - Tipos de dados

- Como identificar o tipo de dados necessário
 - É um numero? Tem casas decimais?

Tipo	Descrição
inteiro	Define variáveis numéricas, destinadas a armazenar números do tipo inteiro (sem casas decimais).
real	Define variáveis numéricas, destinadas a armazenar números inteiros e números do tipo decimal (com casas decimais).

- **Exemplos:**
 - A **idade** deve considerada do tipo dados **inteiro**, porque não tem casas decimais!
 - O **salário** deve ser considerado **real**, uma vez que tem 2 casas decimais para armazenar os centavos.

Pseudocódigo - Tipos de dados

- É um estado?
 - Tem 2 estados? Posso identificar como Verdade (sim) ou Falso (não)?
 - Tem mais do que 2 estados possíveis?

Tipo	Descrição
logico	Define variáveis destinadas a guardar valores lógicos (verdadeiro ou falso)
caracter	Define variáveis destinadas a armazenar apenas um caractere

- **Exemplos:**
 - Para indicarmos se uma pessoa é **casada (ou não)**, devemos considerar o tipo de dados **lógico**, uma vez que podemos armazenar Verdadeiro ou Falso, indicando assim a verdade deste estado lógico.
 - O **género de uma pessoa** (se apenas considerarmos uma letra) deve ser identificado como do tipo **caracter**, uma vez que os valores a armazenar seriam “M” ou “F” - Note-se que apesar de ter apenas 2 estados, não o podemos definir como lógico, uma vez que o tipo de dados lógico apenas permite armazenar o valor “verdadeiro” ou “falso”.

Pseudocódigo - Tipos de dados

- É textual?

Tipo	Descrição
cadeia	Define variáveis destinadas a armazenar texto (considerado também como sendo uma cadeia de caracteres ou strings)

- **Exemplos:**
 - A **morada** deve considerada do tipo dados **cadeia**, porque contem texto!

Pseudocódigo - Tipos de dados

Exercício 15

Identifique o tipo de dados correto:

- Idade
- Casado
- Género
- Altura
- Contribuinte
- Telefone
- Consumo energético
- Saldo bitcoins
- Nome

► Cábula:

Tipo	Descrição
inteiro	Números do tipo inteiro (sem casas decimais).
real	Números do tipo decimal (com casas decimais).
logico	Valores lógicos (verdadeiro ou falso)
caracter	Apenas um caractere
cadeia	Texto (vários caracteres)

Pseudocódigo - Tipos de dados

Exercício 15

Variáveis:

Idade

Casado

Género

Altura

Contribuinte

Telefone

Consumo energético

Saldo em bitcoins

Nome

Definição de necessidades:

0 até 100 (anos)

Sim \ Não

M \ F

0 até 2,99 (metros)

0 até 999999999

0 até 999999999999

0 até 999,9999 (Watts)

0 até 9999,99999999 (bitcoins)

Valor textual

► Cábula:

Tipo	Descrição
inteiro	Números do tipo inteiro (sem casas decimais).
real	Números do tipo decimal (com casas decimais).
logico	Valores lógicos (verdadeiro ou falso)
caracter	Apenas um caractere
cadeia	Texto (vários caracteres)

Pseudocódigo - Tipos de dados

Exercício 15 (solução)

Variáveis:	Definição de necessidades:
Idade	0 até 100 (anos)
Casado	Sim \ Não
Género	M \ F
Altura	0 até 2,99 (metros)
Contribuinte	0 até 999999999
Telefone	0 até 999999999999
Consumo energético	0 até 999,9999 (Watts)
Saldo em bitcoins	0 até 9999,99999999 (bitcoins)
Nome	Valor textual

Tipos de dados:

inteiro (sem casas decimais)

logico (verdade = sim \ falso = não)

caracter (M = masculino \ F = feminino)

real (com casas decimais)

inteiro (sem casas decimais)

inteiro (sem casas decimais)

real (com casas decimais)

real (com casas decimais)

Cadeia (texto)

Pseudocódigo - Distinção de variáveis

- Cada variável deve ter um nome distinto



Pseudocódigo - Distinção de variáveis

- A atribuição de nomes a variáveis obedece a regras:
 - Carateres permitidos:
 - Números (0-9)
 - Letras (a-z/A-Z)
 - Símbolos (são apenas permitidos: “_” e “\$”)
 - O nome das variáveis é sensível a maiúsculas e minúsculas - Inclusive Portugal é Case-Sensitive!
 - O tamanho deve estar compreendido entre 1 a 255 carateres!
 - O primeiro carater não pode ser um número!
 - Não podem existir variáveis com o mesmo nome dentro da mesma estrutura de código!
 - Não pode ser uma palavra reservada.
 - Não são permitidos espaços (exemplo: “nome da pessoa”), em vez disso juntar tudo e separar por maiúsculas (exemplo: “nomeDaPessoa”) ou usar *underscores* (exemplo: “nome_da_pessoa”) - recomenda-se separar por maiúsculas.
 - As variáveis têm que ser declaradas antes de serem utilizadas.

Pseudocódigo - Distinção de variáveis

- Apesar de não ser uma obrigatoriedade explícita, existem alguns requisitos que deverem ser cumpridos:
 - Caracteres com acentuação e o “ç” são permitidos, no entanto não são recomendados!
 - O nome da variável pretende-se com o mínimo de tamanho possível.
 - O nome deve começar por letra minúscula.
 - O nome das variáveis deve identificar devidamente qual o tipo de informação que estas se destinam a armazenar.
 - Idealmente o nome das variáveis deve ser em inglês, uma vez que desta forma estamos a permitir que qualquer outro programador (de um outro país) possa pegar no nosso código, entende-lo e modifica-lo.

Pseudocódigo - Distinção de variáveis

- Palavras reservadas do Portugol
- Grupo de palavras reservadas para o uso exclusivo da linguagem Portugol:

aleatorio	cronometro	escolha	fimenquanto	fimse	leia	outrocaso	repita	vetor
algoritmo	debug	escreva	fimescolha	funcao	limpatela	para	retorne	verdadeiro
arquivo	e	escreval	fimfuncao	inicio	logico	passo	se	xou
ate	eco	faca	fimpara	int	mod	pausa	senao	
caractere	enquanto	falso	fimprocedimento	inteiro	nao	real	timer	
caso	entao	fimalgoritmo	fimrepita	interrompa	ou	procedimento	var	

Pseudocódigo - Distinção de variáveis

Exercício 16

Quais dos seguintes nomes de variáveis são válidos?

- @idade
- idade
- custo%
- _tamanho
- 1cor
- modelo3
- contri buinte
- morada3andar
- contribuinteDoCliente
- género
- ysdfsfsdf
- 5
- €
- real
- Inteiro
- ____morada

Pseudocódigo - Distinção de variáveis

Exercício 16 (solução)

Quais dos seguintes nomes de variáveis são válidos?

- ~~@idade~~ ("@" não é permitido)
- ✓ idade
- ~~custo%~~ ("% "não é permitido)
- _tamanho (Possível, mas não recomendado - deve começar por letra)
- ~~1cor~~ (Não pode começar por numero)
- ✓ modelo3
- ~~contri buinte~~ (Não são permitidos espaços)
- ✓ morada3andar
- ✓ contribuinteDoCliente
- género (Possível, mas não recomendado - acentuação)
- ysdfsfsdf (Possível, mas não recomendado - nome sem sentido)
- ~~5~~ (Não pode começar por numero)
- ~~€~~ ("€" não é permitido)
- ~~real~~ (É uma palavra-reservada)
- ✓ inteiro (Inteiro com "I" maiúsculo não é uma palavra reservada, porque a linguagem é case-sensitive)
- _morada (Possível, mas não recomendado - deve começar por letra)

Declaração de variáveis

Como declarar variáveis no Portugol?



Exemplos:

inteiro	idade;
real	salario;
logico	reformado;
caracter	genero;
cadeia	nome;

Distinção de variáveis

Exercício 17

(Utilizando o Portugol Studio)

Suponha que pretende **guardar os dados de uma encomenda em variáveis**.

Atendendo às regras de atribuição de nomes das variáveis e à seleção ideal do tipos de dados:

- Escolha arbitrariamente e declare 6 variáveis que poderiam estar presentes num software de vendas (desenvolvido em Portugol) de uma qualquer loja à sua escolha.

► Cábula:

Tipo	Descrição
inteiro	Números do tipo inteiro (sem casas decimais).
real	Números do tipo decimal (com casas decimais).
logico	Valores lógicos (verdadeiro ou falso)
caracter	Apenas um caractere
cadeia	Texto (vários caracteres)

Distinção de variáveis

Exercício 17 (solução)

```
inteiro idVenda;  
inteiro quantidade;  
real precoProduto;  
logico comDesconto;  
real totalDesconto  
real precoTotal;
```


Literais - valores que as variáveis podem assumir e como estes devem ser formatados

- A atribuição de valores a variáveis pode ser feita no momento da sua declaração (apelidado de “inicialização de variáveis”) ou pós esta:

- Inicialização de variáveis:

```
inteiro idVenda = 125;  
inteiro quantidade = 2;  
real precoProduto = 25.00;  
logico comDesconto = verdadeiro;  
real totalDesconto = 2.5;  
real precoTotal = 24.37;
```

(Sinal de atribuição)

- Declaração e pós atribuição de valores:

Declaração

```
inteiro idVenda;  
inteiro quantidade;  
real precoProduto;  
logico comDesconto;  
real totalDesconto;  
real precoTotal;
```

Atribuição

```
idVenda = 125;  
quantidade = 2;  
precoProduto = 25.00;  
comDesconto = verdadeiro;  
totalDesconto = 2.5;  
precoTotal = 24.37;
```

Atribuição de valores

Exercício 18

Reaproveitando as variáveis declaradas no Exercício 17, inicialize-as com valores a seu critério - desde que façam sentido perante a função para o qual foram criadas.

Se não tiver no seu exercício variáveis do tipo **inteiro**, **real**, **logico** ou **carater**, crie pelo menos 1 de cada e inicialize-as com valores ao seu critério.

► Cábula:

```
inteiro idVenda = 125;  
inteiro quantidade = 2;  
real precoProduto = 25.00;  
logico comDesconto = verdadeiro;  
real totalDesconto = 2.5;  
real precoTotal = 24.37;
```

Atribuição de valores

Exercício 18 (solução)

```
inteiro idVenda;  
inteiro quantidade;  
real precoProduto;  
logico comDesconto;  
real totalDesconto  
real precoTotal;
```



(solução)

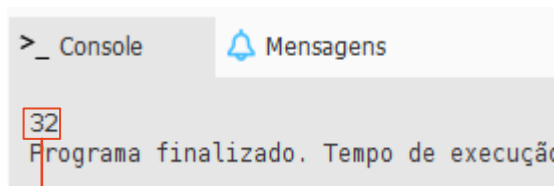
```
inteiro idVenda = 125;  
inteiro quantidade = 2;  
real precoProduto = 25.00;  
logico comDesconto = verdadeiro;  
real totalDesconto = 2.5;  
real precoTotal = 24.37;
```

Atribuição de valores

- O Portugal permite **escrever** informações no ecrã utilizando o comando **escreva()**;

Exemplo:

```
inteiro idade = 32;  
escreva(idade);
```



(Apresenta o valor existente na
variável idade)

Atribuição de valores

Exercício 19

Atendendo ao código abaixo, identifique o que será impresso no ecrã a quando da execução da seguinte ordem de instruções:

```
inteiro nFatura = 125;  
real precoComIva = 25.99;  
escreva(nFatura);
```

Atribuição de valores

Exercício 19 (solução)

```
inteiro nFatura = 125;  
real precoComIva = 25.99;  
escreva(nFatura);
```



(solução)

A screenshot of a console window. The title bar shows a triangle icon, a bell icon, and the text 'Mensagens'. The console content shows the number '125' on the first line and 'Programa finalizado. Tempo de execução: 15 milissegundos' on the second line. The second line is highlighted with a red rectangular box. A red arrow points from the text below to this box.

```
> _ Console Mensagens  
125  
Programa finalizado. Tempo de execução: 15 milissegundos
```

(Esta informação a verde serve somente para indicar o tempo que demorou o nosso programa a ser executado)

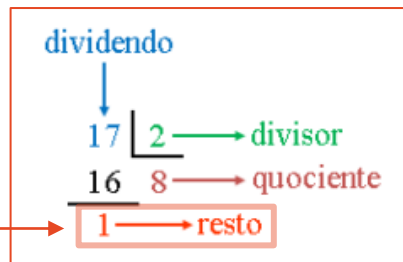
Operações com variáveis - Operadores

Os operadores aritméticos também foram trazidos para a programação, assim, torna-se possível colocar o Portugol a efetuar cálculos.

Operadores aritméticos:

- + Soma
- Subtração
- / Divisão
- * Multiplicação (não confundir e usar o “x”!)
- % Resto da divisão

O que é o resto da divisão?



A saber sobre operações aritméticas

- As operações aritméticas apenas podem ser realizadas entre variáveis numéricas (decimais ou não-decimais) e/ou diretamente com números (com ou sem virgulas)

Operação com números:

```
inteiro resultado = 0;  
resultado = 2 + 2;  
escreva(resultado);
```



```
> _ Console Mensagens  
4  
Programa finalizado. Tempo de execução: 0.00 segundos.
```

Operação com variáveis:

```
inteiro resultado = 0;  
inteiro x = 12;  
inteiro y = 15;  
resultado = x + y;  
escreva(resultado);
```



```
> _ Console Mensagens  
27  
Programa finalizado. Tempo de execução: 0.00 segundos.
```

Operação com variáveis e números:

```
real resultado = 0;  
real y = 2.8;  
resultado = 2 * y;  
escreva(resultado);
```



```
> _ Console Mensagens  
5.6  
Programa finalizado. Tempo de execução: 0.00 segundos.
```


Operações aritméticas

Exercício 20

Inicialize uma variável do tipo “inteiro” (com nome e valor da sua preferência), seguidamente peça ao **Portugol** para apresentar a soma dessa variável pelo valor 56.

► Cábula:

```
real resultado = 0;  
real y = 2.8;  
resultado = 2 * y;  
escreva(resultado);
```

Operações aritméticas

Exercício 20 (solução)

```
inteiro valor = 10;  
inteiro resultado = 0;  
resultado = valor + 56;  
escreva(resultado);
```



```
> _ Console Mensag  
66  
Programa finalizado. T
```

Input do utilizador

Da mesma forma que é possível apresentar informação ao utilizador, também é possível recolhe-la (a partir do teclado).

- Exemplo de como ler do teclado e apresentar no ecrã:

```
inteiro valor = 0;  
leia(valor);  
escreva (valor);
```

Apresenta o que se encontra
na variável “valor”

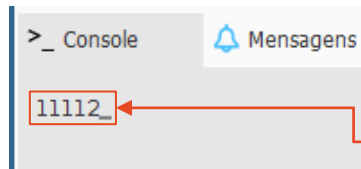
Quando é atingida esta linha, automaticamente o programa fica em pausa, sendo que mais nenhum código é executado até que o utilizador insira algo e prima “Enter” - uma vez esta ação realizada, o código sai de pausa e procede no sentido de realizar as as próximas instruções.

É também nesta linha de código, que é armazenado dentro da variável “valor” o que o utilizador escreveu.

Input do utilizador

- Exemplo de como utilizar a consola no Portugol:

Como usar:

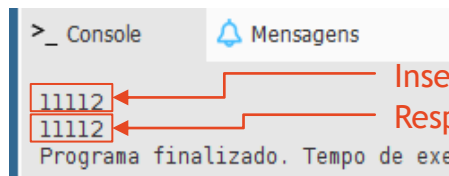


Colocar aqui o cursor de texto e preencher com o teclado (Nota, uma vez que o Portugol está à espera de um número inteiro (uma vez que assim definimos no nosso código) não devem ser colocados valores decimais ou texto!)

Seguidamente apertar o “Enter” do teclado!

Resposta do Portugol:

```
inteiro valor = 0;  
leia(valor);  
escreva (valor);
```



Inserido pelo utilizador

Resposta do Portugol

Operações aritméticas

Exercício 21

Peça dois valores inteiros ao utilizador e no fim apresente a sua soma.

► Cábula:

```
inteiro valor = 0;  
leia(valor);  
escreva (valor);
```

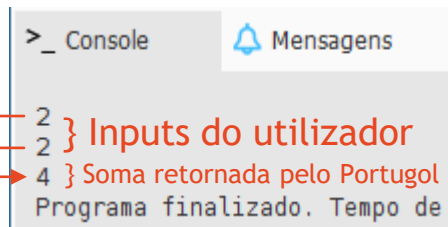
Operações aritméticas

Exercício 21 (solução)

Código:

```
inteiro valor1 = 0;  
inteiro valor2 = 0;  
leia(valor1);  
leia(valor2);  
  
inteiro resultado = valor1 + valor2;  
escreva (resultado);
```

Consola:



The screenshot shows a console window with two tabs: "_ Console" and "Mensagens". The console output is as follows:

```
>_ Console  Mensagens  
2  
2 } Inputs do utilizador  
4 } Soma retornada pelo Portugol  
Programa finalizado. Tempo de
```

Red arrows connect the code to the console output:

- From `leia(valor1);` to the first `2`.
- From `leia(valor2);` to the second `2`.
- From `inteiro resultado = valor1 + valor2;` to the `4`.

Constantes

- As constantes são tratadas como variáveis, no entanto, a quando de definidos os seus valores, estes mantêm-se fixos durante toda a execução do programa e não é possível modifica-los a momento algum!

```
const inteiro CONSTANTE = 255;  
escreva(CONSTANTE);
```

Para declarar uma constante basta apenas iniciar a sua declaração por “const”, tudo o restante que aprendemos até agora sobre variáveis, é também aplicável às constantes, menos a atribuição de valores que não é permitida!

```
> _ Console Mensagens  
255  
Programa finalizado. Tempo c
```

Constantes

Exercício 22

Desenvolva um programa que calcule a área de um círculo em que o utilizador apenas necessita de indicar o raio.

Fórmula:

$$A = \pi * r^2 \quad \text{ou} \quad A = \pi * r * r$$

Nota: Defina π (PI) como uma constante com o valor 3.14159

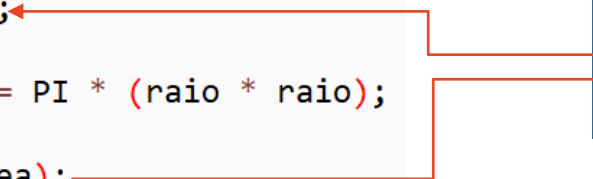
► Cábula:

```
const inteiro CONSTANTE = 255;  
escreva(CONSTANTE);
```


Constantes

Exercício 22 (solução)

```
const real PI = 3.14159;  
real raio;  
leia(raio);  
  
real area = PI * (raio * raio);  
  
escreva(area);
```



```
>_ Console Mensagens  
12  
452.38896  
Programa finalizado. Tempo
```

Verificações

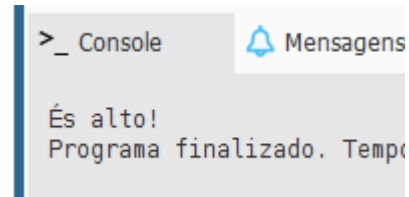
- O Portugol permite-nos fazer verificações, tendo por base a validação de condições.

```
real altura = 1.75;  
  
se (altura < 1.75)  
    escreva ("És baixo!");  
senao  
    escreva ("És alto!");
```

Verificações

- O Portugol permite-nos fazer verificações, tendo por base a validação de condições.

```
real altura = 1.75;  
  
se (altura < 1.75)  
    escreva ("És baixo!");  
senao  
    escreva ("És alto!");
```



Repetições

- Permite-nos ainda repetir determinado grupo de código:

```
para(inteiro i = 0; i <= 5; i++){  
    escreva("Número: " + i + "\n");  
}
```

OU

```
inteiro i = 0;  
enquanto(i < 5){  
    escreva("Número: " + i + "\n");  
    i++;  
}
```



```
> _ Console  Mensag  
Número: 0  
Número: 1  
Número: 2  
Número: 3  
Número: 4  
Número: 5  
  
Programa finalizado. Te
```