

Algoritmos e Modelos de Programação

1º ano / 1º semestre

22,5 horas teórico-práticas - TP (1,5h semanal)

22,5 horas horas práticas laboratoriais - PL (1,5h semanal)

- Engenharia Eletrónica e de Automação (EEA)
- Engenharia Informática (EI)
- Engenharia Mecânica (EM)

Professor: Jorge Simões / Jorge Vieira (turma diurna EM e EEA)

(jsimoes@ispgaya.pt; jvieira@ispgaya.pt)

Conteúdos Inforestudante: <https://inforestudante.ispgaya.pt>

Apresentação da Unidade Curricular

EI

Horas	Segunda	Terça	Quarta
14:30 - 15:00	Análise Matemática I [(JBP)] [4.1 ST]	Algoritmos e Modelos de Programação [(JMS)] [5.1 ST]	Instrumentação e Sistemas Digitais [(JML)] [1.1/1.2 Lab.Electro]
15:00 - 15:30			
15:30 - 16:00			
16:00 - 16:30			
16:30 - 17:00			
17:00 - 17:30			
17:30 - 18:00			
18:00 - 18:30	Análise Matemática I [(MFM)] [4.4 ST/Lab.Inf]	Instrumentação e Sistemas Digitais [(JML)] [2.1 Lab. Redes]	Algoritmos e Modelos de Programação [(JMS)] [5.3 ST/Lab.Inf]
18:30 - 19:00			
19:00 - 19:30			
19:30 - 20:00			
20:00 - 20:30	Algoritmos e Modelos de Programação [(JMS)] [4.1 ST]	Instrumentação e Sistemas Digitais [(JML)] [1.1/1.2 Lab.Electro]	Análise Matemática I [(MFM)] [5.1 ST]
20:30 - 21:00			
21:00 - 21:30			
21:30 - 22:00			
22:00 - 22:30			
22:30 - 23:00			



Aula TP



Aula PL

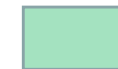
Apresentação da Unidade Curricular

**EM
EEA**

Horas	Segunda	Terça	Quarta	Quinta	Sexta
14:30 - 15:00		Algoritmos e Modelos de Programação [(JMS)] [5.1 ST]	Análise Matemática I [(JMM)] [2.4 Lab. Redes]	Desenho I [(AKF)] [6.2 ST/Lab.Inf]	Álgebra [(JBN)] [5.1 ST]
15:00 - 15:30					
15:30 - 16:00					
16:00 - 16:30	Circuitos Elétricos [(MCS)] [1.3 Lab.MEL]	Análise Matemática I [(JMM)] [4.1 ST]	Algoritmos e Modelos de Programação [(JMS)] [5.3 ST/Lab.Inf]	Desenho I [(AKF)] [5.4 ST/Lab.Inf]	Algoritmos e Modelos de Programação [(JFV)] [5.3 ST/Lab.Inf]
16:30 - 17:00					
17:00 - 17:30					
17:30 - 18:00					
18:00 - 18:30		Circuitos Elétricos [(MCS)] [5.1 ST]	Algoritmos e Modelos de Programação [(JMS)] [5.3 ST/Lab.Inf]	Desenho I [(AKF)] [5.4 ST/Lab.Inf]	Algoritmos e Modelos de Programação [(JFV)] [5.3 ST/Lab.Inf]
18:30 - 19:00					
19:00 - 19:30					
19:30 - 20:00					
20:00 - 20:30	Algoritmos e Modelos de Programação [(JMS)] [4.1 ST]	Circuitos Elétricos [(AMB)] [1.3 Lab.MEL]	Desenho I [(AFC)] [5.4 ST/Lab.Inf]	Análise Matemática I [(MFM)] [4.1 ST]	Álgebra [(JBN)] [5.1 ST]
20:30 - 21:00					
21:00 - 21:30					
21:30 - 22:00	Análise Matemática I [(MFM)] [4.1 ST]	Análise Matemática I [(MFM)] [4.1 ST]	Desenho I [(AFC)] [5.4 ST/Lab.Inf]	Análise Matemática I [(MFM)] [4.1 ST]	Álgebra [(JBN)] [5.1 ST]
22:00 - 22:30					
22:30 - 23:00					



Aula TP



Aula PL

OBJETIVOS

No final da unidade curricular os alunos deverão estar aptos a desenvolver um raciocínio lógico, em termos de conceitos e técnicas de programação, de forma a serem capazes de conceber **algoritmos** para a resolução de problemas práticos de pequena e média complexidade. Os alunos deverão desenvolver as capacidades necessárias para a aplicação dos algoritmos concebidos em programas de computador concretos.

CONHECIMENTOS DE BASE

Conhecimentos de informática na ótica de utilizador. Conhecimentos de Matemática.

PROGRAMA (Resumo)

- Introdução à Algoritmia
- Conceitos de Programação
(fluxogramas, linguagem algorítmica)
- Introdução à Programação com a linguagem Python

BIBLIOGRAFIA ACONSELHADA

- Portela, F., Pereira, T. (2023). Introdução à Algoritmia e Programação com Python, FCA, ISBN: 978-972-722-931-4
- Sobral, S. (2024). Introdução à Programação usando Python, 2ª edição, Edições Sílabo, ISBN: 978-989-561-387-8
- Sobral, S. (2024). Introdução à Programação usando Python – Exercícios Resolvidos, Edições Sílabo, ISBN: 978-989-561-386-1
- Simões, J., Santos, M. (1996), Introdução à Programação, Ed. Catepse, ISBN: 972-97083-0-4, Cota: 004.43/SIMj/INT ISPGaya
- Carvalho, A. (2021). Práticas de Python: Algoritmia e Programação, FCA, 978-972-722-918-5
- Vasconcelos, J. (2015). Python: Algoritmia e Programação Web, FCA, ISBN: 978-972-722-813-3
- Costa, E. (2015). Programação em Python: Fundamentos e Resolução de Problemas, FCA, ISBN: 978-972-722-816-4

LINKS ÚTEIS

- <https://www.w3schools.com/python/>
- <https://docs.python.org/3/tutorial/index.html>
- <https://www.codecademy.com/learn/learn-python-3>
- <https://anandology.com/python-practice-book/>
- outros ...

AVALIAÇÃO POR FREQUÊNCIA

- Trabalhos Práticos (TP1 e TP2), a realizar nas aulas práticas laboratoriais, com um peso de 25% cada.
- Prova Escrita (PE), com um peso de 40%.
- Assiduidade e participação (AP), com um peso de 10%.
- Classificação Final = 25% TP1 + 25% TP2 + 40% PE + 10% AP

AVALIAÇÃO POR EXAME FINAL

- Trabalhos Práticos (TP1 e TP2), com um peso de 25% cada.
- Prova Escrita (PE), com um peso de 50%.
- Classificação Final = 25% TP1 + 25% TP2 + 50% PE
- **Cada aluno deve optar pela avaliação por frequência ou pela avaliação por exame, devendo comunicar a decisão ao regente da unidade curricular até ao fim da primeira semana de aulas. Na ausência de comunicação será entendido que o aluno optou pela avaliação por frequência.**

AVALIAÇÃO EM RECURSO E ÉPOCAS ESPECIAIS

- Trabalhos Práticos (TP1 e TP2), com um peso de 25% cada.
- Prova Escrita (PE), com um peso de 50%.
- Classificação Final = 25% TP1 + 25% TP2 + 50% PE
- **Em todas as avaliações por exame, poderão ser considerados os trabalhos práticos da época de avaliação por frequência caso a classificação obtida nesses trabalhos tenha sido superior a 9,5 valores.**

INSTRUMENTOS DE AVALIAÇÃO

- A Prova Escrita (PE), os Trabalhos Práticos (TP1 e TP2) e a componente de assiduidade e participação (AP) constituem os instrumentos de avaliação da unidade curricular.
- Na Prova Escrita e nos Trabalhos Práticos a **nota mínima é de 7,5** valores, para efeitos de cálculo da classificação final.

**Regulamento de Avaliação de Conhecimentos e Competências
(disponível no Inforestudante)**

AVALIAÇÃO (datas previstas)

- **Trabalhos Práticos** (aula prática-laboratorial)

Trabalho Prático 1: semana de 18 de novembro de 2024

Trabalho Prático 2: semana de 13 de janeiro de 2025

- **Prova Escrita** (teste de escolha múltipla feito na plataforma Moodle)

A marcar pela Direção da Escola

(meio de janeiro / início de fevereiro de 2025)

FUNCIONAMENTO DAS AULAS – 2023/2024

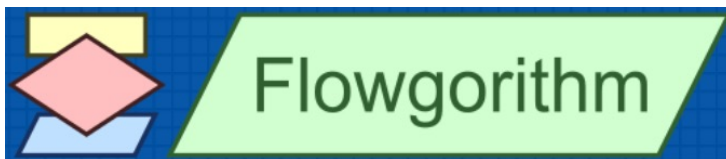
- Aulas Teórico-Práticas: exposição de matéria teórica e exercícios
- Aulas Práticas Laboratoriais: com registo de presenças
 - As aulas práticas-laboratoriais serão ocupadas com a resolução de exercícios práticos (Fluxogramas, Python).
- Os alunos podem resolver os trabalhos práticos usando os computadores do ISPGaya ou usando os seus computadores portáteis.

MATERIAL DE APOIO (a disponibilizar no Inforestudante)

- **Slides das Aulas**
- **Fichas de Exercícios**
- **Livros e links indicados na bibliografia**
- **Software / plataformas de desenvolvimento**

SOFTWARE

- **Flowgorithm** (<http://www.flowgorithm.org>) 
- Portugol Studio (Versão web: <https://portugol-webstudio.cubos.io/>)
- **Python** (<https://www.python.org/downloads/>) 
- Pycharm (<https://www.jetbrains.com/pycharm/>)



**PyCharm Community
Edition**

Algoritmos e Modelos de Programação ?

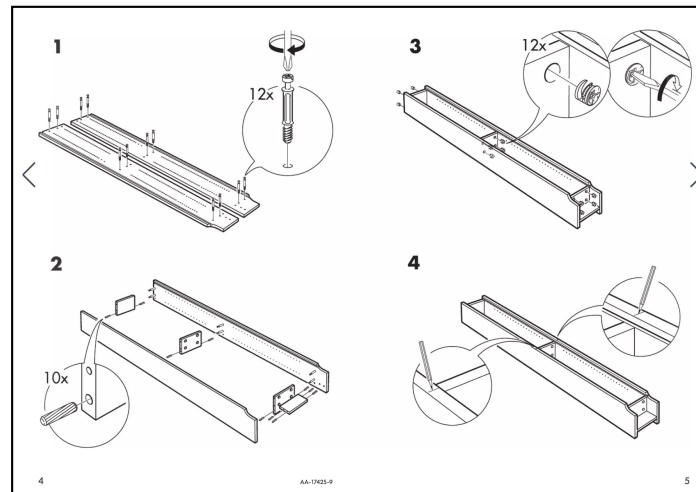
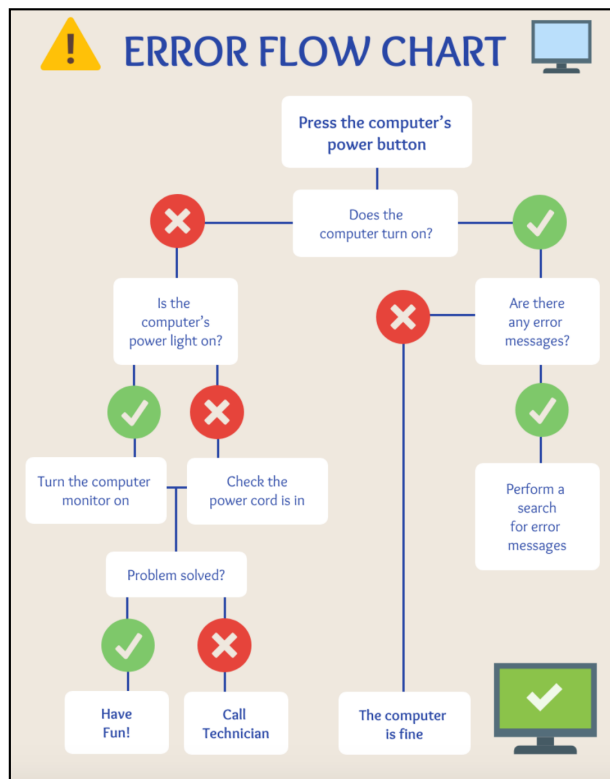
Algoritmo ?

Conjunto finito de instruções, bem definidas e não ambíguas que, ao serem executadas numa determinada ordem, resolvem um problema.



<https://www.humorcomciencia.com/tagtirinha/algoritmo/>

Exemplos de algoritmos?



QUICK TOMATO MOLD
(Pictured below)

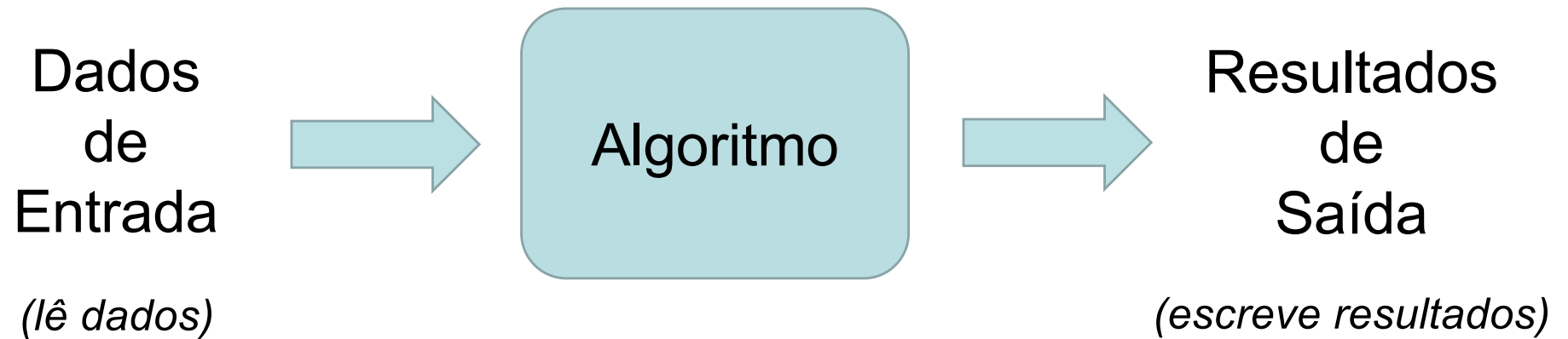
For an unusual appetizer, salad, or relish—a tangy tomato juice mixture that's molded in the juice can.

- 1 can (1 pt. 2 oz.) tomato juice
- 1 package (3 oz.) Jell-O Lemon or Mixed Fruit Gelatin
- 1½ tablespoons lemon juice or vinegar
- ½ teaspoon salt
- Dash of pepper

Bring 1 cup tomato juice to a boil. Stir in Jell-O Gelatin until dissolved. Then pour into juice can, blending with remaining juice. Add seasonings. Chill until firm. To unmold, puncture bottom of can before dipping in warm water—see Tips on Unmolding Jell-O Gelatin on page 82. Slice or serve from a relish dish. Makes 2¾ cups, or 4 to 6 side salads or 8 to 10 relish servings.

Um algoritmo deve ser:

- **Genérico:** deve considerar todos os casos possíveis do problema;
- **Determinístico:** deve ter um comportamento bem definido, não aleatório e sem ambiguidades;
- **Finito:** deve atingir uma solução em tempo finito (em tempo útil) e de forma eficiente;
- **Eficaz:** deve ser capaz de resolver o problema, sem falhas.
- **Capaz de comunicar:** deve ter capacidade de comunicação, recebendo dados (entradas) e apresentando os resultados que obtém (saídas).



Escrever algoritmos para resolver os problemas seguintes:

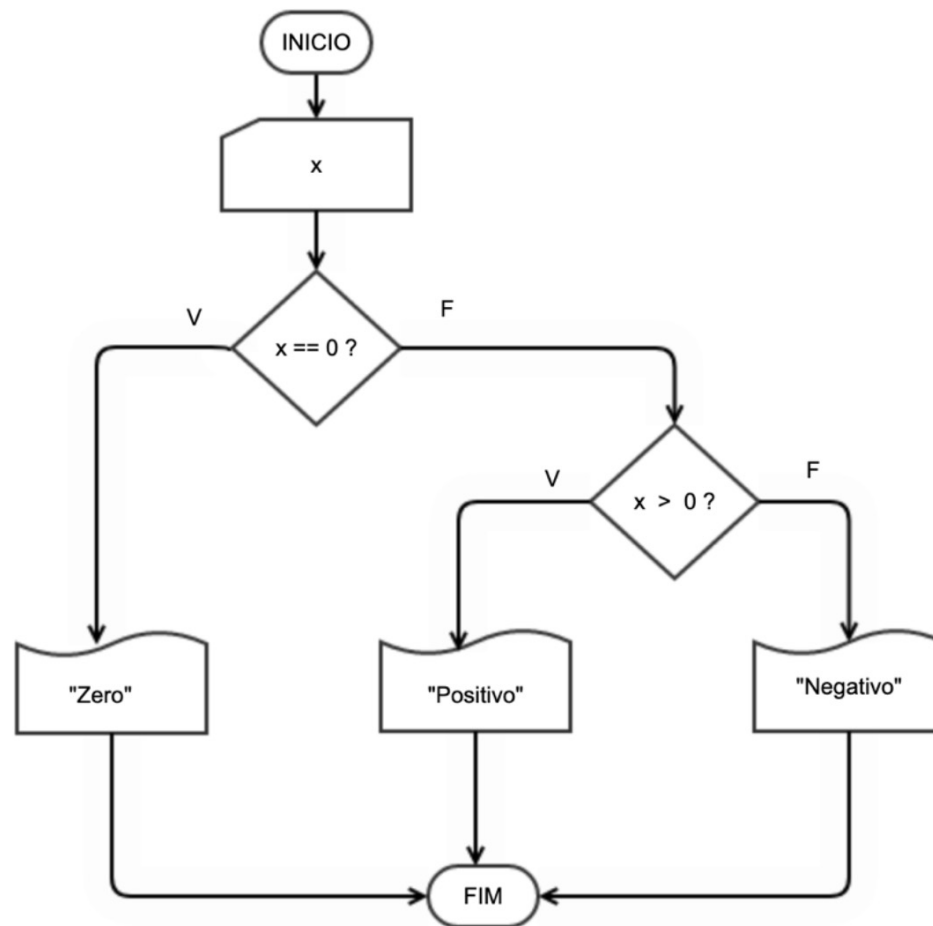
- Mudar um pneu de um automóvel;
- Substituir uma lâmpada fundida de um candeeiro de teto;
- Calcular o máximo divisor comum de dois números inteiros;
- **Verificar se um número inteiro é par ou ímpar;**
- **Verificar se um número inteiro é positivo, negativo ou zero;**
- Verificar se um número inteiro é um número primo;
- Obter o produto de dois números inteiros recorrendo apenas à operação adição;
- **Ler uma sequência de números inteiros que termina com o número 0 e contar quantos dos números lidos são positivos e quantos são negativos.**

Verificar se um número inteiro é par ou ímpar (em linguagem corrente)

1. Ler um número inteiro
2. Dividir o número lido por 2 e guardar o resto dessa divisão
3. Se o resto obtido for igual a zero, então escrever "o número é par"
4. Senão, escrever "o número é ímpar"

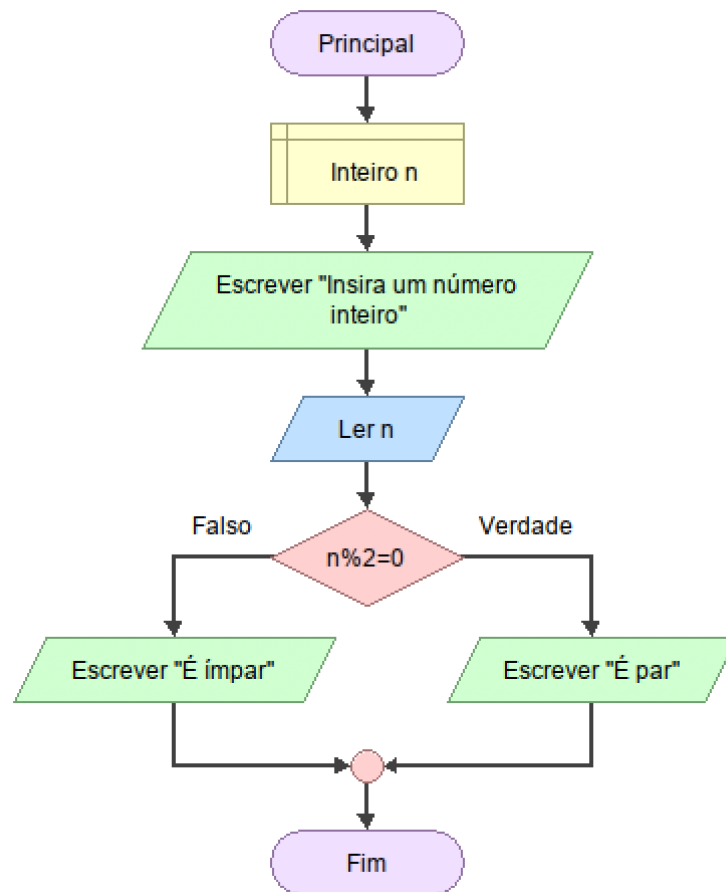
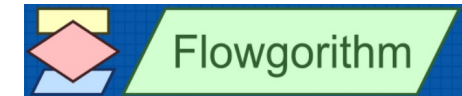
(e se o número lido for 0 ?)

Verificar se um número inteiro é positivo, negativo ou zero (em fluxograma)



- A aplicação **Flowgorithm** permite desenhar fluxogramas (mas com uma notação diferente).
- O **Flowgorithm** permite “executar” o fluxograma e converte-o em instruções numa linguagem de programação à escolha.

Verificar se um número inteiro é par ou ímpar



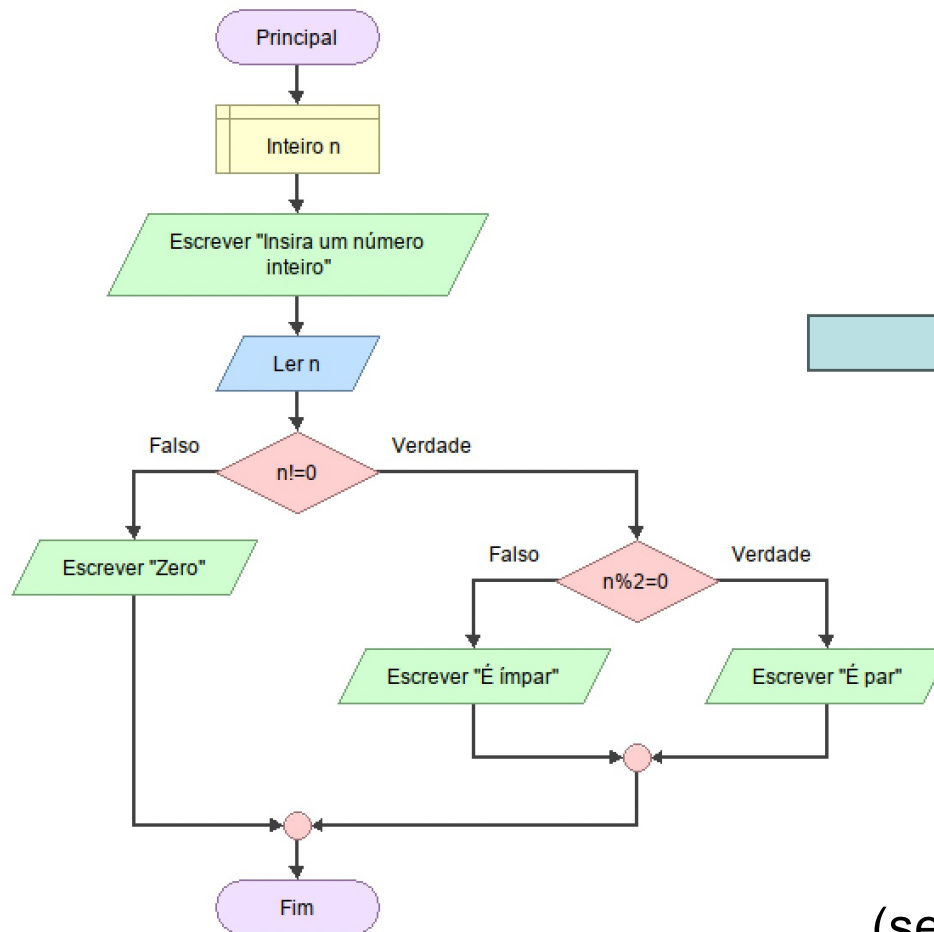
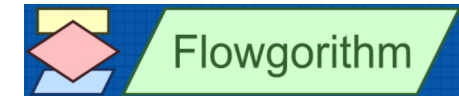
```

print("Insira um número inteiro")
n = int(input())
if n % 2 == 0:
    print("É par")
else:
    print("É ímpar")
  
```

Python

(e se o número lido for 0 ?)

Verificar se um número inteiro é par ou ímpar



```

print("Insira um número inteiro")
n = int(input())
if n != 0:
    if n % 2 == 0:
        print("É par")
    else:
        print("É ímpar")
else:
    print("Zero")
  
```

Python

(se o número lido for 0 escreve "Zero")

Ler uma sequência de números inteiros que termina com o número 0 e contar quantos dos números lidos são positivos e quantos são negativos.

1. Início
2. Iniciar a contagem de positivos em 0 e a contagem de negativos em 0.
3. Ler um número inteiro.
4. Se o número for igual a 0 vai para o passo 8.
5. Senão, se o número for maior que 0 escreva "positivo" e aumenta 1 à contagem de positivos.
6. Senão, se o número for menor que 0 escreva "negativo" aumenta 1 à contagem de negativos.
7. Volta para o passo 3.
8. Escreve a contagem de positivos e a contagem de negativos.
9. Fim

Problema

De um conjunto de 25 moedas de ouro, sabe-se que uma é falsa e pesa menos do que as restantes. Existe disponível uma balança de pratos que, no entanto, apenas pode efetuar um máximo de 3 pesagens. Descubra um algoritmo que permita encontrar a moeda falsa usando a balança de pratos.

