Atividades lp1

Materia	Linguagem de programacao 1
Reviewed	☐
Status	Not started
Data	
Discentes	

Teoricas

Tarefa 15/02

1. Por que é útil para um programador ter alguma experiência no projeto de linguagens, mesmo que ele nunca projete uma linguagem de programação?
    a. Ter experiência no projeto de linguagens pode ajudar um programador a entender melhor as estruturas e princípios subjacentes às linguagens de programação existentes. Isso pode melhorar sua capacidade de escrever código limpo, eficiente e de fácil manutenção, mesmo sem projetar uma linguagem.
2. Como o conhecimento de linguagens de programação pode beneficiar toda a comunidade da computação?
    a. O conhecimento de linguagens de programação beneficia toda a comunidade da computação, pois permite que os programadores se comuniquem de maneira eficaz e escrevam software de qualidade. Isso facilita a colaboração, o compartilhamento de código e a reutilização de soluções, além de promover a inovação e o desenvolvimento de novas tecnologias.
3. Que linguagem de programação tem dominado a Inteligência Artificial nos últimos 50 anos?
    a. Não existe uma única linguagem de programação que tenha dominado a Inteligência Artificial nos últimos 50 anos. Diferentes linguagens foram utilizadas em diferentes contextos e para diferentes finalidades dentro da área de Inteligência Artificial. Alguns exemplos incluem Lisp, Prolog, Python e R.
4. Qual é a desvantagem de ter muitas características em uma linguagem?
    a. A desvantagem de ter muitas características em uma linguagem é que pode aumentar a complexidade e a curva de aprendizado para os programadores. Além disso, um grande número de recursos pode levar a uma maior probabilidade de erros e tornar o código mais difícil de entender e dar manutenção.
5. Como a sobrecarga de operador definida pelo usuário pode prejudicar a legibilidade de um programa?
    a. A sobrecarga de operador definida pelo usuário pode prejudicar a legibilidade de um programa, pois pode introduzir ambiguidades e tornar o código menos intuitivo. Quando os operadores são sobrecarregados com significados diferentes em diferentes contextos, isso pode levar a confusão e dificultar a compreensão do código por outros programadores.

Haskell

Tarefa 16/03

1. Faça um programa que receba duas variáveis: o valor das horas e dos minutos. Em seguida, converta tudo para segundos.

```
main :: IO ()
main = do
  putStrLn "Digite o valor das horas:"
  horas <- readLn
  putStrLn "Digite o valor dos minutos:"
  minutos <- readLn
  let segundos = (horas * 3600) + (minutos * 60)
  putStrLn ("O valor em segundos é: " ++ show segundos)
```

2. Crie um programa que receba a temperatura em graus Celsius e converta para Farenheit.
   Fórmula: F = (ºC * 1,8) + 32

```
main :: IO ()
main = do
  putStrLn "Digite a temperatura em graus Celsius:"
  celsius <- readLn
  let fahrenheit = (celsius * 1.8) + 32
  putStrLn ("A temperatura em Fahrenheit é: " ++ show fahrenheit)
```

3. Faça um programa que pergunte o ano atual e sua idade, em seguida exibe seu ano de nascimento.

```
main :: IO ()
main = do
  putStrLn "Digite o ano atual:"
  anoAtual <- readLn
  putStrLn "Digite a sua idade:"
  idade <- readLn
  let anoNascimento = anoAtual - idade
  putStrLn ("O seu ano de nascimento é: " ++ show anoNascimento)
```

4. Crie um software que recebe o valor do salário e calcula os 27,5% do imposto de renda. A saída do seu programa deve ser o salário bruto (sem abatimento), o valor do imposto a ser pago e o seu salário líquido (após descontar o IR).

```
main :: IO ()
main = do
  putStrLn "Digite o valor do salário:"
  salario <- readLn
  let impostoRenda = salario * 0.275
      salarioLiquido = salario - impostoRenda
  putStrLn ("Salário bruto: " ++ show salario)
  putStrLn ("Imposto de Renda: " ++ show impostoRenda)
  putStrLn ("Salário líquido: " ++ show salarioLiquido)
```

Tarefa 22/03

1. Função `media3` que calcula a média aritmética de três valores e retorna "aprovado" se a média for maior ou igual a 7:

```
media3 :: Double -> Double -> Double -> String
media3 a b c
  | media >= 7 = "Aprovado"
  | otherwise = "Reprovado"
  where
    media = (a + b + c) / 3
```

1. Função `maior2` que retorna o maior valor entre dois valores usando expressões condicionais:

```
maior2 :: Double -> Double -> Double
maior2 a b = if a >= b then a else b
```

1. Função `media2` que retorna o maior valor entre dois valores usando equações com guardas:

```
media2 :: Double -> Double -> Double
media2 a b
  | a >= b = a
  | otherwise = b
```

1. Função `maior3` que retorna o maior valor entre três valores usando expressões condicionais aninhadas e guardas:

```
maior3 :: Double -> Double -> Double -> Double
maior3 a b c = if a >= b
  then if a >= c then a else c
  else if b >= c then b else c
```

ou usando guardas:

```
maior3 :: Double -> Double -> Double -> Double
maior3 a b c
  | a >= b && a >= c = a
  | b >= a && b >= c = b
  | otherwise = c
```

1. Função `maior3` que retorna o maior valor entre três valores sem usar expressões condicionais ou equações com guardas, utilizando a função `maior2` do exercício 3:

```
maior3 :: Double -> Double -> Double -> Double
maior3 a b c = maior2 (maior2 a b) c
```

lista haskell 23/03

1. Faça um algoritmo que leia os valores A, B, C e imprima na tela se a soma de A + B é menor que C.

```
somaMenorC :: Int -> Int -> Int -> String
somaMenorC a b c
| a + b < c = "A + B é menor que C"
| otherwise = "A + B não é menor que C"
```

2. Faça um algoritmo que leia o nome, o sexo e o estado civil de uma pessoa. Caso sexo seja “F” e estado civil seja “CASADA”, solicitar o tempo de casada (anos).

```
dadosPessoa :: String -> Char -> String -> String
dadosPessoa nome sexo estadoCivil
  | sexo == 'F' && estadoCivil == "CASADA" = "Tempo de casada (anos): "
  | otherwise = ""
```

3. Faça um algoritmo para receber um número qualquer e informar na tela se é par ou ímpar.

```
parOuImpar :: Int -> String
parOuImpar n
  | even n = "Par"
  | otherwise = "Ímpar"
```

4. Faça um algoritmo que leia dois valores inteiros A e B se os valores forem iguais deverá se somar os dois, caso contrário multiplique A por B. Ao final de qualquer um dos cálculos deve-se atribuir o resultado para uma variável e mostrar seu conteúdo na tela.

```
calcularResultado :: Int -> Int -> Int
calcularResultado a b
  | a == b = a + b
  | otherwise = a * b
```

5. Encontrar o dobro de um número caso ele seja positivo e o seu triplo caso seja negativo, imprimindo o resultado.

```
dobroOuTriplo :: Int -> Int
dobroOuTriplo n
  | n > 0 = 2 * n
  | n < 0 = 3 * n
  | otherwise = 0
```

6. Escreva um algoritmo que lê dois valores booleanos (lógicos) e então determina se ambos são VERDADEIROS ou FALSOS.

```
verificarBooleanos :: Bool -> Bool -> String
verificarBooleanos b1 b2
  | b1 && b2 = "Ambos são VERDADEIROS"
  | not b1 && not b2 = "Ambos são FALSOS"
  | otherwise = "Um é VERDADEIRO e o outro é FALSO"
```

7. Faça um algoritmo que leia uma variável e some 5 caso seja par ou some 8 caso seja ímpar, imprimir o resultado desta operação.

```
somarParOuImpar :: Int -> Int
somarParOuImpar n
  | even n = n + 5
  | otherwise = n + 8
```

8. Escreva um algoritmo que leia três valores inteiros e diferentes e mostre-os em ordem decrescente.

```
ordemDecrescente :: Int -> Int -> Int -> (Int, Int, Int)
ordemDecrescente a b c
  | a >= b && b >= c = (a, b, c)
  | a >= c && c >= b = (a, c, b)
  | b >= a && a >= c = (b, a, c)
  | b >= c && c >= a = (b, c, a)
  | c >= a && a >= b = (c, a, b)
  | otherwise = (c, b, a)
```

9. Tendo como dados de entrada a altura e o sexo de uma pessoa, construa um algoritmo que calcule seu peso ideal, utilizando as seguintes fórmulas:
   ● para homens: (72.7 * h) – 58;
   ● para mulheres: (62.1 * h) – 44.7.

```
calcularPesoIdeal :: Double -> Char -> Double
calcularPesoIdeal altura sexo
  | sexo == 'M' = (72.7 * altura) - 58
  | sexo == 'F' = (62.1 * altura) - 44.7
  | otherwise = 0.0
```

10. O IMC – Indice de Massa Corporal é um critério da Organização Mundial de Saúde para dar uma indicação sobre a condição de peso de uma pessoa adulta. A fórmula é IMC = peso / (altura)2
    Elabore um algoritmo que leia o peso e a altura de um adulto e mostre sua condição de acordo com a tabela abaixo.
    Condições da IMC em adultos:
    Abaixo de 18,5 Abaixo do peso
    Entre 18,5 e 25 Peso normal
    Entre 25 e 30 Acima do peso
    Acima de 30 obeso

```
calcularIMC :: Double -> Double -> String
calcularIMC peso altura
  | imc < 18.5 = "Abaixo do peso"
  | imc >= 18.5 && imc < 25 = "Peso normal"
  | imc >= 25 && imc < 30 = "Acima do peso"
  | imc >= 30 = "Obeso"
  where
    imc = peso / (altura * altura)
```

11. Elabore um algoritmo que calcule o que deve ser pago por um produto, considerando o preço normal de etiqueta e a escolha da condição de pagamento. Utilize os códigos da tabela a seguir para ler qual a condição de pagamento escolhida e efetuar o cálculo adequado.
    Código condição de pagamento
    a. À vista em dinheiro ou cheque, recebe 10% de desconto
    b. À vista no cartão de crédito, recebe 15% de desconto
    c. Em duas vezes, preço normal de etiqueta sem juros
    d. Em duas vezes, preço normal de etiqueta mais juros de 10%

```
calcularValorPagamento :: Double -> Int -> Double
calcularValorPagamento preco condicao
  | condicao == 1 = preco * 0.9 -- À vista em dinheiro ou cheque (10% de desconto)
  | condicao == 2 = preco * 0.85 -- À vista no cartão de crédito (15% de desconto)
  | condicao == 3 = preco -- Em duas vezes, preço normal de etiqueta sem juros
  | condicao == 4 = preco * 1.1 -- Em duas vezes, preço normal de etiqueta mais juros de 10%
  | otherwise = 0.0 -- Condição inválida

main :: IO ()
main = do
  putStrLn "Informe o preço do produto: "
  precoStr <- getLine
  putStrLn "Informe a condição de pagamento (1 a 4): "
  condicaoStr <- getLine

  let preco = read precoStr :: Double
      condicao = read condicaoStr :: Int
      valorPago = calcularValorPagamento preco condicao

  putStrLn $ "Valor a ser pago: " ++ show valorPago
```

12. Escreva um algoritmo que leia o número de identificação, as 3 notas obtidas por um aluno nas 3 verificações e a média dos exercícios que fazem parte da avaliação, e calcule a média de aproveitamento, usando a fórmula: MA := (nota1 + nota 2 * 2 + nota 3 * 3 + ME)/7. A atribuição dos conceitos obedece a tabela abaixo. O algoritmo deve escrever o número do aluno, suas notas, a média dos exercícios, a média de aproveitamento, o conceito correspondente e a mensagem 'Aprovado' se o conceito for A, B ou C, e 'Reprovado' se o conceito for D ou E. Média de aproveitamento e conceito:

> = 90 A
> = 75 < 90 B
> = 60 < 75 C
> = 40 < 60 D
> < 40 E

```
calcularMediaAproveitamento :: Int -> Double -> Double -> Double -> Double -> IO ()
calcularMediaAproveitamento numeroAluno nota1 nota2 nota3 mediaExercicios = do
  let mediaAproveitamento = (nota1 + nota2 * 2 + nota3 * 3 + mediaExercicios) / 7
      conceito = obterConceito mediaAproveitamento
      status = if conceito `elem` ["A", "B", "C"] then "Aprovado" else "Reprovado"
  putStrLn $ "Número do Aluno: " ++ show numeroAluno
  putStrLn $ "Notas: " ++ show [nota1, nota2, nota3]
  putStrLn $ "Média dos Exercícios: " ++ show mediaExercicios
  putStrLn $ "Média de Aproveitamento: " ++ show mediaAproveitamento
  putStrLn $ "Conceito: " ++ conceito
  putStrLn $ "Status: " ++ status

obterConceito :: Double -> String
obterConceito media
  | media >= 90 = "A"
  | media >= 75 && media < 90 = "B"
  | media >= 60 && media < 75 = "C"
```

```
    | media >= 40 && media < 60 = "D"
    | otherwise = "E"
  ```
```

# Exercicios prolog

## tarefa 12/04

**Exercícios Prolog**
**Base de conhecimento**
progenitor(maria, jose).
progenitor(joao, jose).
progenitor(joao, ana).
progenitor(jose, julia).
progenitor(jose, iris).
progenitor(iris, jorge).

1. Desenhe a árvore genealógica representada pela base de conhecimento.
2. Escreva uma consulta para responder à seguinte pergunta: “Ana é progenitora de Jorge?”
3. Escreva uma consulta para retornar os progenitores de Íris.
4. Escreva uma consulta para retornar os progenitores de José.
5. Escreva uma consulta para retornar todos os pares progenitor/filho da base de conhecimento.
6. Escreva uma consulta para retornar todos os avós de Jorge. Dica: sua consulta será formada por dois termos separados por vírgula.
7. Escreva uma consulta para retornar todos os netos de João.
8. Escreva uma consulta para retornar todos os progenitores comuns de José e Ana.

**respostas:**

Aqui estão as respostas para os exercícios em Prolog, com base na base de conhecimento fornecida:

1. Árvore genealógica:

```
      Maria    João
        |       |
       José    Ana
        / \\
     Julia Iris
           |
         Jorge
```

1. Consulta: "Ana é progenitora de Jorge?"

```
progenitor(ana, jorge).
```

Resposta: False

1. Consulta: Progenitores de Íris

```
progenitor(X, iris).
```

Resposta: X = jose

1. Consulta: Progenitores de José
```

```
progenitor(X, jose).
```

Resposta: X = maria ; X = joao

1. Consulta: Todos os pares progenitor/filho

```
progenitor(X, Y).
```

Resposta: X = maria, Y = jose ; X = joao, Y = jose ; X = joao, Y = ana ; X = jose, Y = julia ; X = jose, Y = iris ; X = iris, Y = jorge

1. Consulta: Todos os avós de Jorge

```
progenitor(X, Y), progenitor(Y, jorge).
```

Resposta: X = jose ; X = maria

1. Consulta: Todos os netos de João

```
progenitor(joao, X), progenitor(X, Y).
```

Resposta: X = jose, Y = julia ; X = jose, Y = iris ; X = iris, Y = jorge

1. Consulta: Todos os progenitores comuns de José e Ana

```
progenitor(X, jose), progenitor(X, ana).
```

Resposta: X = joao

tarefa 19/04

Exercícios Prolog

Base de conhecimento
progenitor(maria, jose).
progenitor(joao, jose).
progenitor(joao, ana).
progenitor(jose, julia).
progenitor(jose, iris).
progenitor(iris, jorge).

1. Desenhe a árvore genealógica representada pela base de conhecimento.
2. Escreva uma consulta para responder à seguinte pergunta: “Ana é progenitora de Jorge?”
3. Escreva uma consulta para retornar os progenitores de Íris.
4. Escreva uma consulta para retornar os progenitores de José.
5. Escreva uma consulta para retornar todos os pares progenitor/filho da base de conhecimento.
6. Escreva uma consulta para retornar todos os avós de Jorge. Dica: sua consulta será formada por dois termos separados por vírgula.
7. Escreva uma consulta para retornar todos os netos de João.
8. Escreva uma consulta para retornar todos os progenitores comuns de José e Ana.
9. Modifique a base de conhecimento para:
   a) Crie a relação de avô e avó
   b) Crie a relação de tio e tia

c) Crie a relação de primo e prima
d) Crie a relação de filho e filha.
e) Crie a relação de Neto e Neta.

Resposta:

1. Árvore genealógica:

```
       Maria    João
          |      |
        José    Ana
         / \\
      Julia Iris
             |
          Jorge
```

1. Consulta: "Ana é progenitora de Jorge?"

```
progenitor(ana, jorge).
```

Resposta: False

1. Consulta: Progenitores de Íris

```
progenitor(X, iris).
```

Resposta: X = jose

1. Consulta: Progenitores de José

```
progenitor(X, jose).
```

Resposta: X = maria ; X = joao

1. Consulta: Todos os pares progenitor/filho

```
progenitor(X, Y).
```

Resposta: X = maria, Y = jose ; X = joao, Y = jose ; X = joao, Y = ana ; X = jose, Y = julia ; X = jose, Y = iris ; X = iris, Y = jorge

1. Consulta: Todos os avós de Jorge

```
progenitor(X, Y), progenitor(Y, jorge).
```

Resposta: X = jose ; X = maria

1. Consulta: Todos os netos de João

```
progenitor(joao, X), progenitor(X, Y).
```

Resposta: X = jose, Y = julia ; X = jose, Y = iris ; X = iris, Y = jorge

1. Consulta: Todos os progenitores comuns de José e Ana

```
progenitor(X, jose), progenitor(X, ana).
```

Resposta: X = joao

1. Modificação da base de conhecimento:

```
progenitor(maria, jose).
progenitor(joao, jose).
progenitor(joao, ana).
progenitor(jose, julia).
progenitor(jose, iris).
progenitor(iris, jorge).

avô(X, Y) :- progenitor(X, Z), progenitor(Z, Y).
avó(X, Y) :- progenitor(X, Z), progenitor(Z, Y).
tio(X, Y) :- progenitor(Z, Y), irmão(X, Z).
tia(X, Y) :- progenitor(Z, Y), irmã(X, Z).
primo(X, Y) :- progenitor(Z, X), progenitor(W, Y), irmão(Z, W).
prima(X, Y) :- progenitor(Z, X), progenitor(W, Y), irmã(Z, W).
filho(X, Y) :- progenitor(Y, X).
filha(X, Y) :- progenitor(Y, X).
neto(X, Y) :- progenitor(Y, Z), progenitor(Z, X).
neta(X, Y) :- progenitor(Y, Z), progenitor(Z, X).

irmão(X, Y) :- progenitor(Z, X), progenitor(Z, Y), X \\= Y.
irmã(X, Y) :- progenitor(Z, X), progenitor(Z, Y), X \\= Y.
```

A base de conhecimento foi modificada para incluir as relações de avô e avó, tio e tia, primo e prima, filho e filha, e neto e neta. Agora é possível fazer consultas relacionadas a essas novas relações familiares.

tarefa dia 20/04

1. Representação de Conhecimento – Autores de Livros. Escreva fatos e/ou regras em Prolog que representem o seguinte conhecimento: Os Maias, livro, Eça de Queiroz, português, inglês, romance, escreveu, autor, nacionalidade, tipo, ficção
   Escreva as seguintes questões em Prolog:
   a) Quem escreveu “Os Maias”?
   b) Que autores portugueses escrevem romances?
   c) Quais os autores de livros de ficção que escreveram livros de outro tipo também?

   ```
   livro(os_maias).
   autor(os_maias, eca_de_queiroz).
   nacionalidade(eca_de_queiroz, portugues).
   tipo(os_maias, romance).
   tipo(os_maias, ficcao).

   escreveu(Autor, Livro) :- autor(Livro, Autor).

   escrevem_romances(Autor) :- autor(Livro, Autor), tipo(Livro, romance), nacionalidade(Autor, portugues).

   escreveram_outro_tipo(Autor) :- autor(Livro1, Autor), autor(Livro2, Autor), tipo(Livro1, Tipo1), tipo(Livro2, Tipo2), Tipo1 \= Tip
   ```

   Agora podemos fazer as seguintes consultas em Prolog:

   a) Quem escreveu "Os Maias"?

   ```prolog
   escreveu(Autor, os_maias).
   ```
   Resposta: Autor = eca_de_queiroz

   b) Que autores portugueses escrevem romances?

   ```prolog
   escrevem_romances(Autor).
   ```
   Resposta: Autor = eca_de_queiroz

   c) Quais os autores de livros de ficção que escreveram livros de outro tipo também?

   ```prolog
   escreveram_outro_tipo(Autor).
   ```
   Resposta: Autor = eca_de_queiroz

2. Representação de Conhecimento – Comidas e Bebidas. Escreva fatos e/ou regras em Prolog que representem o seguinte conhecimento: peru, frango, salmão, solha, cerveja, vinho verde, vinho maduro, Ana, Antonio, Barbara, Bruno, gosta, casado, combina.
   Escreva as seguintes questões em Prolog:

a) Ana e Bruno são casados e gostam de vinho verde?
b) Que bebida combina com salmão?
c) Que comidas combinam com vinho verde?

```
comida(peru).
comida(frango).
comida(salmão).
comida(solha).
bebida(cerveja).
bebida(vinho_verde).
bebida(vinho_maduro).

gosta(ana, vinho_verde).
gosta(bruno, vinho_verde).
casado(ana, bruno).

combina(vinho_verde, salmão).
combina(vinho_maduro, peru).
combina(vinho_maduro, frango).

combinam_com_vinho_verde(Comida) :- combina(vinho_verde, Comida).

casado(ana, bruno), gosta(ana, vinho_verde), gosta(bruno, vinho_verde).
```
Resposta: True

```
combina(Bebida, salmão).
```
Resposta: Bebida = vinho_verde

```
combinam_com_vinho_verde(Comida).
```
Resposta: Comida = salmão

3. Representação de Conhecimento – Desportos e Jogos. Escreva fatos e/ou regras em Prolog que representem o seguinte conhecimento: João, Maria, Ana, casa, cão, xadrez, damas, ténis, natação, apartamento, gato, tigre, homem, mulher, animal, mora_em, gosta_de, jogo, desporto.
   Escreva as seguintes questões em Prolog:
   a) Quem mora num apartamento e gosta de animais?
   b) Será que o João e a Maria moram numa casa e gostam de desportos?
   c) Quem gosta de jogos e de desportos?
   d) Existe alguma mulher que gosta de ténis e gosta de tigres?

```
mora_em(joao, apartamento).
mora_em(maria, casa).
mora_em(ana, casa).

gosta_de(joao, animal).
gosta_de(joao, xadrez).
gosta_de(joao, ténis).
gosta_de(maria, desporto).
gosta_de(maria, casa).
gosta_de(ana, jogos).
gosta_de(ana, desporto).

jogo(xadrez).
jogo(damas).
desporto(ténis).
desporto(natação).

homem(joao).
mulher(maria).
mulher(ana).

gosta_de_jogos_e_desportos(Pessoa) :- gosta_de(Pessoa, jogos), gosta_de(Pessoa, desporto).
```

Agora podemos fazer as seguintes consultas em Prolog:

a) Quem mora num apartamento e gosta de animais?

```prolog
mora_em(Pessoa, apartamento), gosta_de(Pessoa, animal).
```
Resposta: Pessoa = joao

```
b) Será que o João e a Maria moram numa casa e gostam de desportos?

```prolog
mora_em(joao, casa), gosta_de(joao, desporto), mora_em(maria, casa), gosta_de(maria, desporto).
```
Resposta: False

c) Quem gosta de jogos e de desportos?

```prolog
gosta_de_jogos_e_desportos(Pessoa).
```
Resposta: Pessoa = ana

d) Existe alguma mulher que gosta de ténis e gosta de tigres?

mulher(Pessoa), gosta_de(Pessoa, ténis), gosta_de(Pessoa, animal).

Resposta: False
```

4. Representação de Conhecimento – Tweety e Silverster. Traduza as seguintes frases para Prolog:
   “Tweety é um pássaro. Goldie é um peixe. Molie é uma minhoca. Pássaros gostam de minhocas. Gatos gostam de peixes. Gatos gostam de pássaros. Amigos gostam uns dos outros. O meu gato é meu amigo. O meu gato come tudo o que gosta. O meu gato chama-se Silvester.”
   a) Use Prolog para determinar tudo o que come o Silvester?
   b) A resposta é razoável ? Se não for, verifique se o problema está na especificação original ou na sua tradução para Prolog, corrija e execute novamente.

```
passaro(tweety).
peixe(goldie).
minhoca(molie).
gosta(X, Y) :- passaro(X), minhoca(Y).
gosta(gatos, peixe).
gosta(gatos, passaro).
gosta(amigos, amigos).
gato(silvester).
meu_gato(X) :- gato(X), gosta(X, Y), meu_amigo(X).
come(X, Y) :- meu_gato(X), gosta(X, Y).

meu_amigo(silvester).

come(silvester, X).

come(silvester, Comida).
Resposta: Comida = peixe ; Comida = passaro ; Comida = minhoca
```

5. Representação de Conhecimento – Programação e Erros Um estudante acostumado a usar linguagens procedimentais está desenvolvendo um compilador em Prolog. Uma das tarefas consiste em traduzir um código de erro para uma pseudo-descrição em português.
   O código por ele usado é:
   traduza(Codigo, Significado) :- Codigo = 1, Significado = integer_overflow.
   traduza(Codigo, Significado) :- Codigo = 2, Significado = divisao_por_zero.
   traduza(Codigo, Significado) :- Codigo = 3, Significado = id_desconhecido.
   Com sabe, esta não é uma forma apropriada de programar em Prolog. Melhore este código

```
erro(1, integer_overflow).
erro(2, divisao_por_zero).
erro(3, id_desconhecido).

traduza(Codigo, Significado) :- erro(Codigo, Significado).

?- traduza(1, Significado).
Significado = integer_overflow.

?- traduza(2, Significado).
Significado = divisao_por_zero.

?- traduza(3, Significado).
Significado = id_desconhecido.
```

6. Representação de Conhecimento – Cargos e Chefes. Suponha a seguinte base de fatos Prolog:
   cargo(tecnico, rogerio).
   cargo(tecnico, ivone).
   cargo(engenheiro, daniel).
   cargo(engenheiro, isabel).
   cargo(engenheiro, oscar).
   cargo(engenheiro, tomas).
   cargo(engenheiro, ana).
   cargo(supervisor, luis).
   cargo(supervisor_chefe, sonia).
   cargo(secretaria_exec, laura).
   cargo(diretor, santiago).
   chefiado_por(tecnico, engenheiro).
   chefiado_por(engenheiro, supervisor).
   chefiado_por(analista, supervisor).
   chefiado_por(supervisor, supervisor_chefe).
   chefiado_por(supervisor_chefe, director).
   chefiado_por(secretaria_exec, director).
   Onde os predicados cargo/2 e chefiado_por/2 são autoexplicativos. Faça as consultas em Prolog:
   a) ?- chefiado_por(tecnico, X), chefeido_por(X,Y).
   b) ?- chefiado_por(tecnico, X), cargo(X,ivone), cargo(Y,Z).
   c) ?- cargo(supervisor, X); cargo(supervisor, X).
   d) ?- cargo(J,P), (chefiado_por(J, supervisor_chefe); chefiado_por(J, supervisor)).
   e) ?- chefiado_por(P, director), not(cargo(P, carolina)).]

```
a) ?- chefiado_por(tecnico, X), chefiado_por(X, Y).
Resposta: X = engenheiro, Y = supervisor.

b) ?- chefiado_por(tecnico, X), cargo(X, ivone), cargo(Y, Z).
Resposta: X = engenheiro, Y = supervisor, Z = luis.

c) ?- cargo(supervisor, X); cargo(supervisor, X).
Resposta: X = luis; X = sonia.

d) ?- cargo(J, P), (chefiado_por(J, supervisor_chefe); chefiado_por(J, supervisor)).
Resposta: J = tecnico, P = rogerio; J = engenheiro, P = daniel; J = engenheiro, P = isabel; J = engenheiro, P = oscar; J = engenhe

e) ?- chefiado_por(P, director), \+ cargo(P, carolina).
Resposta: true.
```

7. Representação de Conhecimento – Alunos e Professores Considere a seguinte base de fatos exemplo:
   aluno(joao, paradigmas).
   aluno(maria, paradigmas).
   aluno(joel, lab2).
   aluno(joel, estruturas).
   frequenta(joao, feup).
   frequenta(maria, feup).
   frequenta(joel, ist).
   professor(carlos, paradigmas).
   professor(ana_paula, estruturas).
   professor(pedro, lab2). funcionario(pedro, ist).
   funcionario(ana_paula, feup).
   funcionario(carlos, feup).
   Escreva as seguintes regras em prolog:
   a) Quem são os alunos do professor X?
   b) Quem são as pessoas da universidade X? (alunos ou docentes)
   c) Quem é colega de quem? Se aluno: é colega se for colega de disciplina ou colega de curso ou colega de universidade. Se professor: se for professor da mesma universidade.

```
aluno(joao, paradigmas).
aluno(maria, paradigmas).
aluno(joel, lab2).
aluno(joel, estruturas).

frequenta(joao, feup).
frequenta(maria, feup).
frequenta(joel, ist).

professor(carlos, paradigmas).
professor(ana_paula, estruturas).
professor(pedro, lab2).

funcionario(pedro, ist).
funcionario(ana_paula, feup).
funcionario(carlos, feup).

% a) Quem são os alunos do professor X?
aluno_do_professor(Aluno, Professor) :- aluno(Aluno, Disciplina), professor(Professor, Disciplina).

% b) Quem são as pessoas da universidade X? (alunos ou docentes)
pessoa_da_universidade(Pessoa, Universidade) :- (aluno(Pessoa, _); professor(Pessoa, _)), frequenta(Pessoa, Universidade).

% c) Quem é colega de quem?
colega(Pessoa1, Pessoa2) :-
  (aluno(Pessoa1, Disciplina), aluno(Pessoa2, Disciplina), Pessoa1 \= Pessoa2);  % Colega de disciplina
  (aluno(Pessoa1, Curso), aluno(Pessoa2, Curso), Pessoa1 \= Pessoa2);  % Colega de curso
  (frequenta(Pessoa1, Universidade), frequenta(Pessoa2, Universidade), Pessoa1 \= Pessoa2);  % Colega de universidade
  (professor(Pessoa1, Disciplina), professor(Pessoa2, Disciplina), Pessoa1 \= Pessoa2).  % Colega professor da mesma disciplina



a) Quem são os alunos do professor X?
?- aluno_do_professor(Aluno, carlos).
Aluno = joao ;
false.

?- aluno_do_professor(Aluno, ana_paula).
Aluno = joel ;
false.

b) Quem são as pessoas da universidade X? (alunos ou docentes)
?- pessoa_da_universidade(Pessoa, feup).
Pessoa = joao ;
Pessoa = maria ;
Pessoa = ana_paula ;
Pessoa = carlos ;
false.

?- pessoa_da_universidade(Pessoa, ist).
Pessoa = joel ;
Pessoa = pedro ;
false.

c) Quem é colega de quem?
?- colega(joao, maria).
true.

?- colega(joel, maria).
false.

?- colega(joel, pedro).
false.

?- colega(carlos, ana_paula).
false.
```

8. Escreva uma base de conhecimento que represente a cadeia alimentar no reino animal (fatos e regras.

```
/* Fatos */
come(herbivoro, planta).
come(carnivoro, herbivoro).
come(omnivoro, planta).
come(omnivoro, herbivoro).
come(omnivoro, carnivoro).
come(detritivoro, resto_de_comida).

/* Regras */
come(X, Y) :- come(X, Z), come(Z, Y).
```

```
/* Exemplos */
come(herbivoro, X).
come(carnivoro, X).
come(omnivoro, X).
come(detritivoro, X).
```