

## **Lista de Exercícios 3 – PGENE503 – Sistemas de Tempo Real**

Discente: Ana Cristina da Silva Vieira.

Docente: Drº Lucas Cordeiro

1) Ada termina cada construtor com `end <nome do construtor>`; C não usa um marcador final. Quais são as vantagens e desvantagens dos projetos destas linguagens?

**Resposta:**

- **Vantagens do Ada:** o uso de `end <nome>` aumenta a legibilidade e reduz erros, pois o programador sabe exatamente qual bloco está sendo finalizado — útil em programas grandes e com muitos níveis de aninhamento.
- **Desvantagens do Ada:** torna o código mais verboso e menos conciso, exigindo mais digitação.
- **Vantagens do C:** sintaxe mais simples e curta, o que agiliza a escrita de programas pequenos.
- **Desvantagens do C:** menor legibilidade e maior probabilidade de erros em estruturas aninhadas, pois não há indicação explícita de qual bloco o `}` fecha.

2) Java e C são case-sensitive; Ada não é. Quais são os argumentos a favor e contra do case-sensitive?

**Resposta:**

- **A favor:** permite criar identificadores diferentes com nomes semelhantes (ex.: `valor` e `Valor`), útil em convenções de código e distinções semânticas.
- **Contra:** aumenta a chance de erros de digitação difíceis de detectar (ex.: `Total ≠ total`), além de reduzir a clareza para iniciantes.

- **Ada (não case-sensitive):** promove legibilidade e consistência, mas impede diferenciação sutil de nomes e reduz a flexibilidade.

3) Uma linguagem deve sempre solicitar que sejam dados valores iniciais para as variáveis? Cite pelo menos um problema relacionado à não inicialização de variáveis usadas no programa.

**Resposta:: valor e Valor: valor e Valor**

Sim, idealmente a linguagem deve exigir inicialização para evitar comportamentos indefinidos.

**Problema:** em C, uma variável local não inicializada contém “lixo de memória”; usá-la em cálculos ou condições pode causar erros lógicos ou falhas graves (ex.: divisão por valor aleatório, falha de segurança).

4) O uso do comando `exit` em Ada leva a programas legíveis e confiáveis?

**Resposta:**

Depende do contexto. O `exit` pode melhorar a legibilidade quando usado para sair de loops em condições claras, evitando blocos aninhados.

Por outro lado, o uso excessivo pode dificultar o rastreamento do fluxo do programa, reduzindo a confiabilidade.

Portanto, deve ser usado **com parcimônia e clareza de propósito**.

5) Escreva programas em Ada95 para:

Abaixo estão exemplos sintéticos e corretos de cada item (resumidos para clareza):

**a) Inverter string**

```
with Ada.Text_IO; use Ada.Text_IO;
procedure Inverte_String is
    S : String := "ANA";
begin
    for I in reverse S'Range loop
        Put(S(I));
    end loop;
end Inverte_String;
```

IMAGEM DA EXECUÇÃO DO CÓDIGO NO ONECOMPILER

The screenshot shows the OneCompiler IDE interface. In the top left, it says "HelloWorld.adb". The top right has navigation links: AI, NEW, ADA (selected), RUN, and LOGIN. The main area contains Ada code to reverse a string "THIAGO". The output window shows the reversed string "OGAIHT".

```

1 with Ada.Text_IO; use Ada.Text_IO;
2 procedure Inverte_String is
3     S : String := "THIAGO";
4 begin
5     for I in reverse S'Range loop
6         Put(S(I));
7     end loop;
8 end Inverte_String;
9

```

Output:  
OGAIHT

### b) Soma de vetores A e B

```

with Ada.Text_IO; use Ada.Text_IO;

procedure Soma_Vetores is
    A : array (1 .. 3) of Integer := (1, 2, 3);
    B : array (1 .. 3) of Integer := (4, 5, 6);
    C : array (1 .. 3) of Integer;
begin
    Put_Line("===== SOMA DE VETORES =====");
    Put_Line("");
    Put_Line("Vetor A: (1, 2, 3)");
    Put_Line("Vetor B: (4, 5, 6)");
    Put_Line("");

    Put_Line("Resultado da soma elemento a elemento:");
    for I in A'Range loop
        C(I) := A(I) + B(I);
        Put_Line(" " & Integer'Image(A(I)) & " +
                & Integer'Image(B(I)) & " =
                & Integer'Image(C(I)));
    end loop;

    Put_Line("");
    Put_Line("Vetor resultante C: (" &
            Integer'Image(C(1)) & ", " &
            Integer'Image(C(2)) & ", " &
            Integer'Image(C(3)) & " )");

    Put_Line("=====");
end Soma_Vetores;

```

### IMAGEM DA EXECUÇÃO DO CÓDIGO

The screenshot shows the OneCompiler IDE interface. The left pane displays the Ada source code for a program named `HelloWorld.adb`. The right pane shows the execution results. The output window has sections for `STDIN`, `Input for the program (Optional)`, and `Output`. The `Output` section contains the program's output, which includes the sum of two vectors and the resulting vector.

```

1 with Ada.Text_IO; use Ada.Text_IO;
2
3 procedure Soma_Vetores is
4   A : array (1 .. 3) of Integer := (1, 2, 3);
5   B : array (1 .. 3) of Integer := (4, 5, 6);
6   C : array (1 .. 3) of Integer;
7 begin
8   Put_Line("===== SOMA DE VETORES =====");
9   Put_Line("");
10  Put_Line("Vetor A: (1, 2, 3)");
11  Put_Line("Vetor B: (4, 5, 6)");
12  Put_Line("");
13
14  Put_Line("Resultado da soma elemento a elemento:");
15  for I in A'Range loop
16    C(I) := A(I) + B(I);
17    Put_Line(" " & Integer'Image(A(I)) & " +
18      & Integer'Image(B(I)) & " =
19      & Integer'Image(C(I)));
20  end loop;
21
22  Put_Line("");
23  Put_Line("Vetor resultante C: (" &
24    Integer'Image(C(1)) & ", " &
25    Integer'Image(C(2)) & ", " &
26    Integer'Image(C(3)) & ")");
27
28  Put_Line("===== ");
29 end Soma_Vetores;
30
31

```

Output:

```

===== SOMA DE VETORES =====
Vetor A: (1, 2, 3)
Vetor B: (4, 5, 6)

Resultado da soma elemento a elemento:
 1 + 4 = 5
 2 + 5 = 7
 3 + 6 = 9

Vetor resultante C: ( 5, 7, 9 )
===== 
```

### c) Contar vogais

```

with Ada.Text_IO; use Ada.Text_IO;

procedure Conta_Vogais is
  Vogais : constant String := "AEIOUaeiou";
  Texto : String := "Ada é uma linguagem de programação de última " &
    "geração que equipes de desenvolvimento no mundo todo estão usando" &
    "para softwares críticos: desde microkernels e sistemas embarcados" &
    "de pequeno porte e em tempo real até aplicativos corporativos de" &
    "larga escala, e tudo o que há entre esses dois extremos.";
  Cont_Vogais : Integer := 0;
begin
  Put_Line("===== ");
  Put_Line("          ANALISADOR DE TEXTO");
  Put_Line("===== ");
  New_Line;

  Put_Line("Texto analisado:");
  Put_Line(Texto);
  New_Line;

  for I in Texto'Range loop
    for V of Vogais loop
      if Texto(I) = V then
        Cont_Vogais := Cont_Vogais + 1;
        exit;
      end if;
    end loop;
  end loop;

  Put_Line("-----"); 
```

```

Put_Line("Total de vogais encontradas: " & Integer'Image(Cont_Vogais));
Put_Line("=====");
end Conta_Vogais;

```

## IMAGEM DA EXECUÇÃO

The screenshot shows the OneCompiler interface with the following details:

- Code Area:** The code is written in Ada. It includes a procedure `Conta_Vogais` that reads a string from the user, counts the vowels, and prints the total count.
- Output Area:**
  - Text input: "Ada é uma linguagem de programação de última geração que equilibra o uso de recursos no mundo todo entre usos de baixa e alta sofisticação: desde microkernels e sistemas embarcados de pequeno porte e em tempo real até aplicativos corporativos de larga escala, e tudo o que há entre esses dois extremos."
  - Output text:  
-----  
ANALISADOR DE TEXTO  
-----  
Texto analisado:  
Ada é uma linguagem de programação de ultima geração que equipes de desenvolvimento todo mundo todo entre usos de baixa e alta sofisticação: desde microkernels e sistemas embarcados de pequeno porte e em tempo real até aplicativos corporativos de larga escala, e tudo o que há entre esses dois extremos.  
-----  
Total de vogais encontradas: 188  
-----

d) Lido um string, escrever quantas e quais são as vogais.

```

with Ada.Text_IO; use Ada.Text_IO;
with Ada.Strings.Unbounded; use Ada.Strings.Unbounded;

procedure Solucao_D is
    Entrada      : String := Get_Line;
    Contador      : Natural := 0;
    Vogais_Encontradas : Unbounded_String := To_Unbounded_String("");
begin
    Put_Line("-- Solucao D: Contar e Listar Vogais ---");
    Put("Digite uma string e pressione Enter: ");

    for I in Entrada'Range loop
        -- Converte o caractere para maiúsculo para simplificar a comparação
        case Character'Val(Character'Pos(Entrada(I)) - 32) is
            when 'A' / 'E' / 'I' / 'O' / 'U' =>
                Contador := Contador + 1;
                Append(Vogais_Encontradas, Entrada(I));
            when others =>
                null;
        end case;
    end loop;

    New_Line;
    Put_Line("Numero de vogais: " & Contador'Img);
    Put_Line("Vogais encontradas: " & To_String(Vogais_Encontradas));
end Solucao_D;

```

## IMAGEM DA EXECUÇÃO

The screenshot shows the OneCompiler IDE interface. In the top navigation bar, there are links for Pricing, Learn, Code, Deploy, and More. Below the navigation bar, there are buttons for AI, NEW, ADA, and RUN. The ADA button is highlighted.

The main area displays Ada code for a program named `HelloWorld.adb`. The code is as follows:

```

1 with Ada.Text_IO; use Ada.Text_IO;
2 with Ada.Strings.Unbounded; use Ada.Strings.Unbounded;
3
4 procedure Solucao_D is
5   Entrada : String := Get_Line;
6   Contador : Natural := 0;
7   Vogais_Encontradas : Unbounded_String := To_Unbounded_String("");
8 begin
9   Put_Line("--- Solucao D: Contar e Listar Vogais ---");
10  Put("Digite uma string e pressione Enter: ");
11
12  for I in Entrada'Range loop
13    -- Converte o caractere para maiúsculo para simplificar a comparação
14    case Character'Val(Character'Pos(Entrada(I)) - 32) is
15      when 'A' | 'E' | 'I' | 'O' | 'U' =>
16        Contador := Contador + 1;
17        Append(Vogais_Encontradas, Entrada(I));
18      when others =>
19        null;
20    end case;
21  end loop;
22
23  New_Line;
24  Put_Line("Numero de vogais: " & Contador'Img);
25  Put_Line("Vogais encontradas: " & To_String(Vogais_Encontradas));
26 end Solucao_D;

```

In the right panel, under the `STDIN` section, the user has typed `Ana Cristina`. Under the `Output` section, the program's response is shown:

```

--- Solucao D: Contar e Listar Vogais ---
Digite uma string e pressione Enter:
Numero de vogais: 4
Vogais encontradas: aiia

```

- e) O produto interno entre dois vetores A e B é o escalar obtido por  $\sum a_i x_{bi}$  onde n é o tamanho dos dois vetores.

```

with Ada.Text_IO;      use Ada.Text_IO;
with Ada.Integer_Text_IO; use Ada.Integer_Text_IO;

procedure Solucao_E is
  N : Positive;
begin
  Put_Line("--- Solucao E: Produto Interno de Vetores ---");
  Put("Digite o tamanho dos vetores: ");
  Get(N);

  declare
    type Vetor is array (1 .. N) of Integer;
    A, B      : Vetor;
    Produto_Interno : Integer := 0;
  begin
    Put_Line("Digite os" & N'Img & " elementos do vetor A:");
    for I in A'Range loop
      Get(A(I));
    end loop;
    Skip_Line;

    Put_Line("Digite os" & N'Img & " elementos do vetor B:");
    for I in B'Range loop
      Get(B(I));
    end loop;
    Skip_Line;

    for I in A'Range loop

```

```

    Produto_Interno := Produto_Interno + (A(I) * B(I));
end loop;

New_Line;
Put_Line("O Produto Interno é: " & Produto_Interno'Img);
end;
end Solucao_E;

```

## IMAGEM DA EXECUÇÃO

The screenshot shows the OneCompiler IDE interface. The code is named HelloWorld.adb. The code itself is as follows:

```

1 with Ada.Text_IO;           use Ada.Text_IO;
2 with Ada.Integer_Text_IO;   use Ada.Integer_Text_IO;
3
4 procedure Solucao_E is
5     N : Positive;
6 begin
7     Put_Line("-- Solucao E: Produto Interno de Vetores ---");
8     Put("Digite o tamanho dos vetores: ");
9     Get(N);
10
11    declare
12        type Vetor is array (1 .. N) of Integer;
13        A : array (1 .. N) of Integer;
14        B : array (1 .. N) of Integer;
15        Produto_Interno : Integer := 0;
16    begin
17        Put_Line("Digite os" & N'Img & " elementos do vetor A:");
18        for I in A'Range loop
19            Get(A(I));
20        end loop;
21        Skip_Line;
22
23        Put_Line("Digite os" & N'Img & " elementos do vetor B:");
24        for I in B'Range loop
25            Get(B(I));
26        end loop;
27        Skip_Line;
28
29        for I in A'Range loop
30            Produto_Interno := Produto_Interno + (A(I) * B(I));
31        end loop;
32
33        New_Line;
34        Put_Line("O Produto Interno é: " & Produto_Interno'Img);
35    end;
36 end Solucao_E;

```

The STDIN pane shows the user input: 2, followed by 10 10, and then 12. The Output pane shows the program's response: --- Solucao E: Produto Interno de Vetores ---. Then it prompts for vector sizes, receives 2, then 10 10, and 12. Finally, it outputs the result: O Produto Interno é: 30.

f) Lidos dois strings A e B, verifique se o string B está contido em A.

```

with Ada.Text_IO; use Ada.Text_IO;

procedure Solucao_F is
    String_A : String := Get_Line;
    String_B : String := Get_Line;
    Contido : Boolean := False;
begin
    Put_Line("--- Solucao F: Verificar se String B está contida em A ---");
    Put("Digite a string A (maior) e pressione Enter: ");
    Put("Digite a string B (menor) e pressione Enter: ");
    New_Line;

    if String_B'Length <= String_A'Length and String_B'Length > 0 then
        for I in String_A'First .. (String_A'Last - String_B'Length + 1) loop
            if String_A(I .. I + String_B'Length - 1) = String_B then
                Contido := True;
                exit; -- Sai do loop assim que encontrar
            end if;
        end loop;
    end if;

    if Contido then
        Put_Line("Resultado: A string B ESTA contida na string A.");
    else

```

```

Put_Line("Resultado: A string B NAO ESTA contida na string A.");
end if;
end Solucao_F;

```

## IMAGEM DA EXECUÇÃO

The screenshot shows the OneCompiler interface with the following details:

- Title Bar:** OneCompiler
- File:** HelloWorld.adb
- Code Area:**

```

1 with Ada.Text_IO; use Ada.Text_IO;
2
3 procedure Solucao_F is
4   String_A : String := Get_Line;
5   String_B : String := Get_Line;
6   Contido : Boolean := False;
7 begin
8   Put_Line("--- Solucao F: Verificar se String B está contida em A ---");
9   Put("Digite a string A (maior) e pressione Enter: ");
10  Put("Digite a string B (menor) e pressione Enter: ");
11  New_Line;
12
13  if String_B'Length <= String_A'Length and String_B'Length > 0 then
14    for I in String_A'First .. (String_A'Last - String_B'Length + 1) loop
15      if String_A(I..I + String_B'Length - 1) = String_B then
16        Contido := True;
17        exit;
18      end if;
19    end loop;
20  end if;
21
22  if Contido then
23    Put_Line("Resultado: A string B ESTA contida na string A.");
24  else
25    Put_Line("Resultado: A string B NAO ESTA contida na string A.");
26  end if;
27 end Solucao_F;

```
- Output Area (STDOUT):**

Ada é uma linguagem de programação de última geração que equipes de desenvolvimento no mundo todo estão usando para softwares críticos: desde microkernels e sistemas embarcados de pequeno porte e em tempo real até aplicativos corporativos de larga escala, e tudo o que há entre esses dois extremos.
- Output Area (STDIN):**

--- Solucao F: Verificar se String B está contida em A ---  
Digite a string A (maior) e pressione Enter: Digite a string B (menor)  
Resultado: A string B ESTA contida na string A.

g) Lido um array numérico de n elementos, ache o maior.

```

with Ada.Text_IO;      use Ada.Text_IO;
with Ada.Integer_Text_IO; use Ada.Integer_Text_IO;

procedure Solucao_G is
  N : Positive;
begin
  Put_Line("--- Solucao G: Encontrar o Maior Elemento do Array ---");
  Put("Digite a quantidade de elementos do array: ");
  Get(N);

  declare
    type Vetor is array(1..N) of Integer;
    V   : Vetor;
    Maior : Integer;
  begin
    Put_Line("Digite os" & N'Img & "elementos:");
    for I in V'Range loop
      Get(V(I));
    end loop;

    Maior := V(V'First); -- Assume que o primeiro é o maior inicialmente
    for I in V'Range loop
      if V(I) > Maior then
        Maior := V(I);
      end if;
    end loop;

    New_Line;
    Put_Line("O maior elemento é: " & Maior'Img);
  end;

```

```
end Solucao_G;
```

### IMAGEM DA EXECUÇÃO

The screenshot shows the OneCompiler interface with the code for 'HelloWorld.adb' and the question 'Questão 5:g) Maior Elemento de um Array'. The code is an Ada program that reads the size of an array and its elements from standard input, then finds and prints the maximum element. The standard input shows the user entering '7' followed by the array elements '10 5 9 15 3 4 5 0'. The standard output shows the program's response: '--- Solucao G: Encontrar o Maior Elemento do Array ---', 'Digite a quantidade de elementos do array: Digte os 7 elementos:', and 'O maior elemento é: 50'.

```

1 with Ada.Text_IO;      use Ada.Text_IO;
2 with Ada.Integer_Text_IO; use Ada.Integer_Text_IO;
3
4 procedure Solucao_G is
5   N : Positive;
6 begin
7   Put_Line("--- Solucao G: Encontrar o Maior Elemento do Array ---");
8   Put("Digite a quantidade de elementos do array: ");
9   Get(N);
10
11  declare
12    type Vetor is array(1..N) of Integer;
13    V : Vetor;
14    Maior : Integer;
15 begin
16   Put_Line("Digite os" & N'Img & " elementos:");
17   for I in V'Range loop
18     Get(V(I));
19   end loop;
20
21   Maior := V(V'First); -- Assume que o primeiro é o maior inicialmente
22   for I in V'Range loop
23     if V(I) > Maior then
24       Maior := V(I);
25     end if;
26   end loop;
27
28   New_Line;
29   Put_Line("O maior elemento é: " & Maior'Img);
30 end;
31 end Solucao_G;

```

h) Lido um conjunto de n elementos, ordená-lo crescentemente.

```

with Ada.Text_IO;      use Ada.Text_IO;
with Ada.Integer_Text_IO; use Ada.Integer_Text_IO;

procedure Solucao_H is
  N : Positive;
begin
  Put_Line("--- Solucao H: Ordenar Array (Bubble Sort) ---");
  Put("Digite a quantidade de elementos a ordenar: ");
  Get(N);

  declare
    type Vetor is array(1..N) of Integer;
    V : Vetor;
    Temp : Integer;
  begin
    Put_Line("Digite os" & N'Img & " elementos:");
    for I in V'Range loop Get(V(I)); end loop;

    -- Algoritmo Bubble Sort
    for I in V'First .. V'Last - 1 loop
      for J in V'First .. V'Last - I loop
        if V(J) > V(J+1) then
          Temp := V(J);
          V(J) := V(J+1);
          V(J+1) := Temp;
        end if;
      end loop;
    end loop;

    New_Line;
    Put("Array ordenado: ");
    for I in V'Range loop
      Put(Integer'Image(V(I)));
    end loop;
    New_Line;
  end;

```

```

end;
end Solucao_H;

```

## IMAGEM DA EXECUÇÃO

```

HelloWorld.adb  Questão 5: h) Ordenação de Array

1 with Ada.Text_IO;      use Ada.Text_IO;
2 with Ada.Integer_Text_IO; use Ada.Integer_Text_IO;
3
4 procedure Solucao_H is
5   N : Positive;
6 begin
7   Put_Line("--- Solucao H: Ordenar Array (Bubble Sort) ---");
8   Put("Digite a quantidade de elementos a ordenar: ");
9   Get(N);
10
11  declare
12    type Vetor is array(1..N) of Integer;
13    V : Vetor;
14    Temp : Integer;
15  begin
16    Put_Line("Digite os" & N'Img & " elementos:");
17    for I in V'Range loop Get(V(I)); end loop;
18
19    Algorithm_Bubble_Sort:
20    for I in V'First .. V'Last - 1 loop
21      for J in V'First .. V'Last - I loop
22        if V(J) > V(J+1) then
23          Temp := V(J);
24          V(J) := V(J+1);
25          V(J+1) := Temp;
26        end if;
27      end loop;
28    end loop;
29
30    New_Line;
31    Put("Array ordenado: ");
32    for I in V'Range loop
33      Put(Integer'Image(V(I)));
34    end loop;
35    New_Line;
36  end;
37 end Solucao_H;

```

i) Leia uma matriz A e calcule a média aritmética de seus elementos.

```

with Ada.Text_IO;      use Ada.Text_IO;
with Ada.Integer_Text_IO; use Ada.Integer_Text_IO;
with Ada.Float_Text_IO; use Ada.Float_Text_IO;

procedure Solucao_I is
  Linhas, Colunas : Positive;
begin
  Put_Line("--- Solucao I: Média dos Elementos de uma Matriz ---");
  Put("Digite o numero de linhas: "); Get(Linhas);
  Put("Digite o numero de colunas: "); Get(Colunas);

  declare
    type Matriz is array (1 .. Linhas, 1 .. Colunas) of Integer;
    A : Matriz;
    Soma : Long_Integer := 0;
    Media : Float;
  begin
    Put_Line("Digite os elementos da matriz:");
    for I in A'Range(1) loop
      for J in A'Range(2) loop
        Get(A(I,J));
        -- AQUI ESTÁ A CORREÇÃO --
        Soma := Soma + Long_Integer(A(I,J));
      end loop;
    end loop;

    Media := Float(Soma) / Float(Linhas * Colunas);
    New_Line;
    Put("A média dos elementos é: ");
    Put(Media, Fore=>2, Aft=>2, Exp=>0);
    New_Line;
  end;
end Solucao_I;

```

## IMAGEM DA EXECUÇÃO

The screenshot shows the OneCompiler IDE interface. The code editor contains a file named `HelloWorld.adb` with the following Ada code:

```

1  with Ada.Text_IO;      use Ada.Text_IO;
2  with Ada.Integer_Text_IO; use Ada.Integer_Text_IO;
3  with Ada.Float_Text_IO; use Ada.Float_Text_IO;
4
5  procedure Solucao_I is
6    Linhas, Colunas : Positive;
7  begin
8    Put_Line("--- Solucao I: Média dos Elementos de uma Matriz ---");
9    Put("Digite o numero de linhas: "); Get(Linhas);
10   Put("Digite o numero de colunas: "); Get(Colunas);
11
12  declare
13    type Matriz is array (1 .. Linhas, 1 .. Colunas) of Integer;
14    A : Matriz;
15    Soma : Long_Integer := 0;
16    Media : Float;
17
18  begin
19    Put_Line("Digite os elementos da matriz:");
20    for I in A'Range(1) loop
21      for J in A'Range(2) loop
22        Get(A(I,J));
23        -- AQUI ESTÁ A CORREÇÃO --
24        Soma := Soma + Long_Integer(A(I,J));
25      end loop;
26    end loop;
27
28    Media := Float(Soma) / Float(Linhas * Colunas);
29    New_Line;
30    Put("A média dos elementos é: ");
31    Put(Media, Fore => 2, Aft => 2, Exp => 0);
32    New_Line;
33  end Solucao_I;

```

The output window shows the following interaction:

```

2
3
738
591
Output:
--- Solucao I: Média dos Elementos de uma Matriz ---
Digite o numero de linhas: Digite o numero de colunas: Digite os elem
A média dos elementos é: 5.50

```

j) Lidas duas matrizes A, de dimensões mxn e B, de dimensões pxq, calcule a matriz C, soma de A e B.

```

with Ada.Text_IO;      use Ada.Text_IO;
with Ada.Integer_Text_IO; use Ada.Integer_Text_IO;

procedure Solucao_J is
  Linhas, Colunas : Positive;
begin
  Put_Line("--- Solucao J: Soma de Matrizes ---");
  Put("Digite o numero de linhas das matrizes: "); Get(Linhas);
  Put("Digite o numero de colunas das matrizes: "); Get(Colunas);

  declare
    type Matriz is array (1 .. Linhas, 1 .. Colunas) of Integer;
    A, B, C : Matriz;
  begin
    Put_Line("Digite os elementos da matriz A:");
    for I in A'Range(1) loop for J in A'Range(2) loop Get(A(I,J)); end loop; end loop;

    Put_Line("Digite os elementos da matriz B:");
    for I in B'Range(1) loop for J in B'Range(2) loop Get(B(I,J)); end loop; end loop;

    for I in C'Range(1) loop
      for J in C'Range(2) loop
        C(I,J) := A(I,J) + B(I,J);
      end loop;
    end loop;

    New_Line;
    Put_Line("Matriz C (soma):");
    for I in C'Range(1) loop
      for J in C'Range(2) loop Put(Item => C(I,J), Width => 5); end loop;
      New_Line;
    end loop;
  end;
end Solucao_J;

```

## IMAGEM DA EXECUÇÃO

```

HelloWorld.adb                                     Questão 5;j) Soma de Matrizes
1  with Ada.Text_IO;      use Ada.Text_IO;
2  with Ada.Integer_Text_IO; use Ada.Integer_Text_IO;
3
4  procedure Solucao_J is
5    Linhas, Colunas : Positive;
6  begin
7    Put_Line("--- Solucao J: Soma de Matrizes ---");
8    Put("Digite o numero de linhas das matrizes: "); Get(Linhas);
9    Put("Digite o numero de colunas das matrizes: "); Get(Colunas);
10
11   declare
12     type Matriz is array (1 .. Linhas, 1 .. Colunas) of Integer;
13     A, B, C : Matriz;
14   begin
15     Put_Line("Digite os elementos da matriz A:");
16     for I in A'Range() loop for J in A'Range(2) loop Get(A(I,J)); end loop; end loop;
17
18     Put_Line("Digite os elementos da matriz B:");
19     for I in B'Range() loop for J in B'Range(2) loop Get(B(I,J)); end loop; end loop;
20
21     for I in C'Range() loop
22       for J in C'Range() loop
23         C(I,J) := A(I,J) + B(I,J);
24       end loop;
25     end loop;
26
27     New_Line;
28     Put_Line("Matriz C (soma):");
29     for I in C'Range() loop
30       for J in C'Range(2) loop Put(Item => C(I,J), Width => 5); end loop;
31       New_Line;
32     end loop;
33   end;
34 end Solucao_J;

```

STDIN:

```

2
3
123
789
456
321

```

Output:

```

--- Solucao J: Soma de Matrizes ---
Digite o numero de linhas das matrizes: Digite o numero de colunas das
Digite os elementos da matriz B:
Matriz C (soma):
      5    7    9
     10   10   10

```

- k) Lida uma matriz, gere sua matriz transposta. Dica: Matriz transposta, em matemática, é o resultado da troca de linhas por colunas em uma determinada matriz. A matriz transposta de uma matriz qualquer M é representada por  $M^T$ .

```

with Ada.Text_IO;      use Ada.Text_IO;
with Ada.Integer_Text_IO; use Ada.Integer_Text_IO;

procedure Solucao_K is
  Linhas, Colunas : Positive;
begin
  Put_Line("--- Solucao K: Matriz Transposta ---");
  Put("Digite o numero de linhas da matriz: "); Get(Linhas);
  Put("Digite o numero de colunas da matriz: "); Get(Colunas);

  declare
    type Matriz is array (1 .. Linhas, 1 .. Colunas) of Integer;
    type Matriz_T is array (1 .. Colunas, 1 .. Linhas) of Integer;
    A           : Matriz;
    A_Transposta : Matriz_T; -- NOME ALTERADO AQUI
  begin
    Put_Line("Digite os elementos da matriz original:");
    for I in A'Range(1) loop
      for J in A'Range(2) loop
        Get(A(I,J));
      end loop;
    end loop;

    -- Loop de transposição
    for I in A'Range(1) loop
      for J in A'Range(2) loop
        A_Transposta(J,I) := A(I,J); -- NOME ALTERADO AQUI
      end loop;
    end loop;
  end;
end Solucao_K;

```

```

New_Line;
Put_Line("Matriz Transposta:");
-- Loop para imprimir a matriz transposta
for I in A_Transposta'Range(1) loop
    for J in A_Transposta'Range(2) loop
        Put(Item => A_Transposta(I,J), Width => 5); -- NOME ALTERADO AQUI
    end loop;
    New_Line;
end loop;
end;
end Solucao_K;

```

## IMAGEM DA EXECUÇÃO

The screenshot shows the OneCompiler IDE interface. The code in the editor is identical to the one above. In the 'Output' pane, the user inputs '2' (number of rows) and '3' (number of columns). The program then outputs the transposed matrix:

```

--- Solucao K: Matriz Transposta ---
Digite o numero de linhas da matriz: Digite o numero de colunas da matriz:
Matriz Transposta:
4 9
5 8
6 7

```

- I) Lidas duas matrizes A, de dimensões mxn e B, de dimensões pxq, calcule a matriz C, produto de A e B.

```

with Ada.Text_IO;      use Ada.Text_IO;
with Ada.Integer_Text_IO; use Ada.Integer_Text_IO;

procedure Solucao_L is
    M, N, P : Positive;
begin
    Put_Line("--- Solucao L: Produto de Matrizes (A[m,n] * B[n,p]) ---");
    Put("Digite as linhas da Matriz A (m): "); Get(M);
    Put("Digite as colunas de A / linhas de B (n): "); Get(N);
    Put("Digite as colunas da Matriz B (p): "); Get(P);

    declare
        type Matriz_A is array (1..M, 1..N) of Integer;
        type Matriz_B is array (1..N, 1..P) of Integer;
        type Matriz_C is array (1..M, 1..P) of Integer;
        A : Matriz_A;

```

```

B : Matriz_B;
C : Matriz_C;
begin
  Put_Line("Digite os elementos da matriz A:");
  for I in A'Range(1) loop for J in A'Range(2) loop Get(A(I,J)); end loop; end loop;

  Put_Line("Digite os elementos da matriz B:");
  for I in B'Range(1) loop for J in B'Range(2) loop Get(B(I,J)); end loop; end loop;

  for I in C'Range(1) loop
    for J in C'Range(2) loop
      C(I,J) := 0;
      for K in A'Range(2) loop
        C(I,J) := C(I,J) + A(I,K) * B(K,J);
      end loop;
    end loop;
  end loop;

  New_Line;
  Put_Line("Matriz C (Produto):");
  for I in C'Range(1) loop
    for J in C'Range(2) loop Put(Item => C(I,J), Width => 6); end loop;
    New_Line;
  end loop;
end;
end Solucao_L;

```

## IMAGEM DA EXECUÇÃO

The screenshot shows the AdaIDE environment with the code for matrix multiplication. The code uses Ada.Text\_IO and Ada.Integer\_Text\_IO packages. It defines three matrices (A, B, C) and their dimensions (M, N, P). The program prompts the user to input the elements of matrices A and B. It then calculates the product matrix C = A \* B. Finally, it prints the resulting matrix C.

```

HelloWorld.adb  Questão 5: I) Produto de Matrizes
1  with Ada.Text_IO;      use Ada.Text_IO;
2  with Ada.Integer_Text_IO; use Ada.Integer_Text_IO;
3
4  procedure Solucao_L is
5    M, N, P : Positive;
6  begin
7    Put_Line("-- Solucao L: Produto de Matrizes (A[m,n] * B[n,p]) ---");
8    Put("Digite as linhas da Matriz A (m):"); Get(N);
9    Put("Digite as colunas de A / linhas de B (n):"); Get(N);
10   Put("Digite as colunas da Matriz B (p):"); Get(P);
11
12  declare
13    type Matriz_A is array (1..M, 1..N) of Integer;
14    type Matriz_B is array (1..N, 1..P) of Integer;
15    type Matriz_C is array (1..M, 1..P) of Integer;
16    A : Matriz_A;
17    B : Matriz_B;
18    C : Matriz_C;
19  begin
20    Put_Line("Digite os elementos da matriz A:");
21    for I in A'Range(1) loop for J in A'Range(2) loop Get(A(I,J)); end loop; end loop;
22
23    Put_Line("Digite os elementos da matriz B:");
24    for I in B'Range(1) loop for J in B'Range(2) loop Get(B(I,J)); end loop; end loop;
25
26    for I in C'Range(1) loop
27      for J in C'Range(2) loop
28        C(I,J) := 0;
29        for K in A'Range(2) loop
30          C(I,J) := C(I,J) + A(I,K) * B(K,J);
31        end loop;
32      end loop;
33    end loop;
34
35    New_Line;
36    Put_Line("Matriz C (Produto):");
37    for I in C'Range(1) loop
38      for J in C'Range(2) loop Put(Item => C(I,J), Width => 6); end loop;
39      New_Line;
40    end loop;
41  end;
42 end Solucao_L;

```

The right pane shows the standard input (STDIN) and standard output (STDOUT). The user inputs the dimensions M=3, N=2, P=1. The program then prints the resulting matrix C:

```

123
10515
852
235

```

```

-- Solucao L: Produto de Matrizes (A[m,n] * B[n,p]) ---
Digite as linhas da Matriz A (m): Digite as colunas de A / linhas de B
Digite os elementos da matriz B:
160 90 65

```

m) Lida uma matriz quadrada, calcule a somatória dos elementos da diagonal principal.

```

with Ada.Text_IO;      use Ada.Text_IO;
with Ada.Integer_Text_IO; use Ada.Integer_Text_IO;

procedure Solucao_M is
    N : Positive;
begin
    Put_Line("--- Solucao M: Soma da Diagonal Principal ---");
    Put("Digite a ordem da matriz quadrada (N): ");
    Get(N);

    declare
        type Matriz_Q is array(1..N, 1..N) of Integer;
        A : Matriz_Q;
        Soma_Diag : Integer := 0;
    begin
        Put_Line("Digite os elementos da matriz:");
        for I in A'Range(1) loop for J in A'Range(2) loop Get(A(I,J)); end loop; end loop;

        for I in A'Range(1) loop
            Soma_Diag := Soma_Diag + A(I,I);
        end loop;

        New_Line;
        Put_Line("A soma da diagonal principal é: " & Soma_Diag'Img);
    end;
end Solucao_M;

```

### IMAGEM DA EXECUÇÃO

```

HelloWorld.adb                                     Questão 5:m) Soma da Diagonal Principal
1  with Ada.Text_IO;      use Ada.Text_IO;
2  with Ada.Integer_Text_IO; use Ada.Integer_Text_IO;
3
4  procedure Solucao_M is
5      N : Positive;
6  begin
7      Put_Line("--- Solucao M: Soma da Diagonal Principal ---");
8      Put("Digite a ordem da matriz quadrada (N): ");
9      Get(N);

10     declare
11         type Matriz_Q is array(1..N, 1..N) of Integer;
12         A : Matriz_Q;
13         Soma_Diag : Integer := 0;
14     begin
15         Put_Line("Digite os elementos da matriz:");
16         for I in A'Range(1) loop for J in A'Range(2) loop Get(A(I,J)); end loop; end loop;

17         for I in A'Range(1) loop
18             Soma_Diag := Soma_Diag + A(I,I);
19         end loop;

20         New_Line;
21         Put_Line("A soma da diagonal principal é: " & Soma_Diag'Img);
22     end;
23 end Solucao_M;

```

STDIN

3  
235  
751  
340

Output:

--- Solucao M: Soma da Diagonal Principal ---  
Digite a ordem da matriz quadrada (N): Digite os elementos da matriz  
A soma da diagonal principal é: 7

n) Lidas duas matrizes de tamanho 3x3 verifique se uma é inversa da outra.

```

with Ada.Text_IO;      use Ada.Text_IO;
with Ada.Float_Text_IO; use Ada.Float_Text_IO;

procedure Solucao_N is
    N : constant Integer := 3;
    type Matriz is array(1..N, 1..N) of Float;
    A, B, Produto : Matriz;
    Eh_Inversa : Boolean := True;
    Epsilon : constant Float := 0.01; -- Tolerância para erros de ponto flutuante
begin
    Put_Line("--- Solucao N: Verificar Matriz Inversa (3x3 ---");
    Put_Line("Digite os elementos da matriz A (use . como separador decimal):");
    for I in A'Range(1) loop for J in A'Range(2) loop Get(A(I,J)); end loop; end loop;

    Put_Line("Digite os elementos da matriz B:");
    for I in B'Range(1) loop for J in B'Range(2) loop Get(B(I,J)); end loop; end loop;

    for I in Produto'Range(1) loop
        for J in Produto'Range(2) loop
            Produto(I,J) := 0.0;
            for K in A'Range(2) loop
                Produto(I,J) := Produto(I,J) + A(I,K) * B(K,J);
            end loop;
            end loop;
        end loop;

    for I in Produto'Range(1) loop
        for J in Produto'Range(2) loop
            if I = J then
                if abs(Produto(I,J) - 1.0) > Epsilon then Eh_Inversa := False; exit; end if;
            else
                if abs(Produto(I,J)) > Epsilon then Eh_Inversa := False; exit; end if;
            end if;
            end loop;
            if not Eh_Inversa then exit; end if;
        end loop;
    end loop;

    New_Line;
    if Eh_Inversa then
        Put_Line("Resultado: As matrizes SÃO inversas uma da outra.");
    else
        Put_Line("Resultado: As matrizes NÃO SÃO inversas.");
    end if;
end Solucao_N;

```

## IMAGEM DA EXECUÇÃO

The screenshot shows the OneCompiler IDE interface. The code in the editor is:

```

3
4
5 procedure Solucao_N is
6   N : constant Integer := 3;
7   type Matriz is array(1..N, 1..N) of Float;
8   A, B, Produto : Matriz;
9   Eh_Inversa : Boolean := True;
10  Epsilon : constant Float := 0.0001; -- Tolerância para erros de ponto flutuante
11 begin
12  Put_Line("... Solucao N: Verificar Matriz Inversa (3x3) ...");
13  Put_Line("Digite os elementos da matriz A (use . como separador decimal):");
14  for I in A'Range() loop for J in A'Range() loop Get(A(I,J)); end loop; end loop;
15  Put_Line("Digite os elementos da matriz B:"); 
16  for I in B'Range() loop for J in B'Range() loop Get(B(I,J)); end loop; end loop;
17  for I in Produto'Range() loop
18    for J in Produto'Range() loop
19      Produto(I,J) := 0.0;
20    end loop;
21    for K in A'Range(2) loop
22      Produto(I,J) := Produto(I,J) + A(I,K) * B(K,J);
23      end loop;
24    end loop;
25  end loop;
26
27  for I in Produto'Range() loop
28    for J in Produto'Range() loop
29      if I = J then
30        if abs(Produto(I,J)) - 1.0 > Epsilon then Eh_Inversa := False; exit; end if;
31        else
32          if abs(Produto(I,J)) > Epsilon then Eh_Inversa := False; exit; end if;
33        end if;
34      end loop;
35      if not Eh_Inversa then exit; end if;
36    end loop;
37
38  New_Line;
39  if Eh_Inversa then
40    Put_Line("Resultado: As matrizes SÃO inversas uma da outra.");
41  else
42    Put_Line("Resultado: As matrizes NÃO SÃO inversas.");
43  end if;
44 end Solucao_N;

```

The output window shows the input provided by the user:

```

56.4
98.2
32.1
39.6
78.9
25.8

```

And the resulting output:

```

Output:
--- Solucao N: Verificar Matriz Inversa (3x3) ...
Digite os elementos da matriz A (use . como separador decimal):
Digite os elementos da matriz B:
Resultado: As matrizes NÃO SÃO inversas.

```

As imagens ilustram os dois cenários de validação do algoritmo. A primeira imagem (acima) demonstra um teste com resultado negativo, onde o programa identifica corretamente que as matrizes de entrada **não** são inversas. Em contraste, a segunda imagem (abaixo) mostra uma execução bem-sucedida, confirmando que o segundo par de matrizes fornecido é inverso um do outro.

## IMAGEM 2 DA EXECUÇÃO

The screenshot shows the OneCompiler IDE interface. The code in the editor is identical to the one in the previous screenshot:

```

3
4
5 procedure Solucao_N is
6   N : constant Integer := 3;
7   type Matriz is array(1..N, 1..N) of Float;
8   A, B, Produto : Matriz;
9   Eh_Inversa : Boolean := True;
10  Epsilon : constant Float := 0.01; -- Tolerância para erros de ponto flutuante
11 begin
12  Put_Line("... Solucao N: Verificar Matriz Inversa (3x3) ...");
13  Put_Line("Digite os elementos da matriz A (use . como separador decimal):");
14  for I in A'Range() loop for J in A'Range() loop Get(A(I,J)); end loop; end loop;
15  Put_Line("Digite os elementos da matriz B:"); 
16  for I in B'Range() loop for J in B'Range() loop Get(B(I,J)); end loop; end loop;
17  for I in Produto'Range() loop
18    for J in Produto'Range() loop
19      Produto(I,J) := 0.0;
20    end loop;
21    for K in A'Range(2) loop
22      Produto(I,J) := Produto(I,J) + A(I,K) * B(K,J);
23      end loop;
24    end loop;
25  end loop;
26
27  for I in Produto'Range() loop
28    for J in Produto'Range() loop
29      if I = J then
30        if abs(Produto(I,J)) - 1.0 > Epsilon then Eh_Inversa := False; exit; end if;
31        else
32          if abs(Produto(I,J)) > Epsilon then Eh_Inversa := False; exit; end if;
33        end if;
34      end loop;
35      if not Eh_Inversa then exit; end if;
36    end loop;
37
38  New_Line;
39  if Eh_Inversa then
40    Put_Line("Resultado: As matrizes SÃO inversas uma da outra.");
41  else
42    Put_Line("Resultado: As matrizes NÃO SÃO inversas.");
43  end if;
44 end Solucao_N;

```

The output window shows the input provided by the user:

```

2.0 0.0 1.0
3.0 0.0 0.0
5.0 1.0 1.0
0.0 0.333 0.0
-1.0 -1.0 1.0
1.0 -0.667 1.0

```

And the resulting output:

```

Output:
--- Solucao N: Verificar Matriz Inversa (3x3) ...
Digite os elementos da matriz A (use . como separador decimal):
Digite os elementos da matriz B:
Resultado: As matrizes SÃO inversas uma da outra.

```

- o) Lida uma matriz, calcule a porcentagem de elementos nulos desta matriz.

```

with Ada.Text_IO;      use Ada.Text_IO;
with Ada.Integer_Text_IO; use Ada.Integer_Text_IO;
with Ada.Float_Text_IO; use Ada.Float_Text_IO;

```

```

procedure Solucao_0 is
    Linhas, Colunas : Positive;
begin
    Put_Line("--- Solucao 0: Percentual de Elementos Nulos na Matriz ---");
    Put("Digite o numero de linhas: "); Get(Linhas);
    Put("Digite o numero de colunas: "); Get(Colunas);

declare
    type Matriz is array (1 .. Linhas, 1 .. Colunas) of Integer;
    A          : Matriz;
    Contador_Nulos : Natural := 0;
    Percentual   : Float;
begin
    Put_Line("Digite os elementos da matriz:");
    for I in A'Range(1) loop
        for J in A'Range(2) loop
            Get(A(I,J));
            if A(I,J) = 0 then
                Contador_Nulos := Contador_Nulos + 1;
            end if;
        end loop;
    end loop;

    Percentual := (Float(Contador_Nulos) / Float(Linhas * Colunas)) * 100.0;
    New_Line;
    Put("O percentual de elementos nulos é: ");
    Put(Percentual, Fore=>2, Aft=>2, Exp=>0);
    Put_Line("%");
end;
end Solucao_0;

```

## IMAGEM DA EXECUÇÃO

The screenshot shows the OneCompiler IDE interface. The code editor contains the Ada code for Solucao\_0. The run tab is active, and the output window displays the following:

```

STDIN
3
4
5 8.00
0.940
0.036

Output:
--- Solucao 0: Percentual de Elementos Nulos na Matriz ---
Digite o numero de linhas: Digite o numero de colunas: Digite os elementos da matriz:
0 percentual de elementos nulos é: 50.00 %

```

- 6) Implemente um procedimento que imprime a área de qualquer objeto derivado de Object.

### Resposta conceitual:

Cria-se um tipo base `Object` com método `Area`. Cada tipo derivado (Círculo, Retângulo etc.) sobrescreve `Area`. O procedimento genérico imprime o valor da área.

```

with Ada.Text_IO;
with Ada.Float_Text_IO;
with Ada.Numerics; -- Para usar o valor de Pi
-- Não precisamos de Ada.Tags

procedure Main is

  -- Pacote Geometria (Especificação)
  package Geometria is

    type Object is abstract tagged null record;
    function Area (Item : in Object) return Float is abstract;
    procedure Imprime_Area (Item : in Object'Class);

    type Circulo is new Object with record
      Raio : Float;
    end record;
    overriding function Area (C : in Circulo) return Float;

    type Retangulo is new Object with record
      Largura, Altura : Float;
    end record;
    overriding function Area (R : in Retangulo) return Float;

  end Geometria;

  -- Pacote Geometria (Corpo)
  package body Geometria is

    overriding function Area (C : in Circulo) return Float is
    begin
      return Ada.Numerics.Pi * (C.Raio ** 2);
    end Area;

    overriding function Area (R : in Retangulo) return Float is
    begin
      return R.Largura * R.Altura;
    end Area;

    procedure Imprime_Area (Item : in Object'Class) is
      Nome_Do_Tipo : String(1..9);
    begin

      if Item in Circulo'Class then
        Nome_Do_Tipo := "Circulo ";
      elsif Item in Retangulo'Class then
        Nome_Do_Tipo := "Retangulo";
      else
        Nome_Do_Tipo := "Objeto ";
      end if;

      Ada.Text_IO.Put ("A área do " & Nome_Do_Tipo & " é: ");
      Ada.Float_Text_IO.Put (Item => Area(Item), Fore => 1, Aft => 2, Exp =>
0);
      Ada.Text_IO.New_Line;
    end Imprime_Area;

```

```

end Geometria;

use Geometria;

-Aqui eu coloco os valores nas variaveis

Meu_Circulo : Circulo := (Raio => 2.0);
Meu_Retangulo : Retangulo := (Largura => 2.0, Altura => 10.0);

begin -- Início da parte executável do procedimento Main

Ada.Text_IO.Put_Line("--- Testando o procedimento polimórfico ---");

Imprime_Area (Meu_Circulo);
Imprime_Area (Meu_Retangulo);

end Main;

```

### IMAGEM DA EXECUÇÃO

```

HelloWorld.adb
Questão 6

1 with Ada.Text_IO;
2 with Ada.Float_Text_IO;
3 with Ada.Numerics; -- Para usar o valor de Pi
4 -- Não precisamos de Ada.Tags
5
6 procedure Main is
7   -- Recete Geometria (Especificação)
8   package Geometria is
9     type Object is abstract tagged null record;
10    function Area (Item : in Object) return Float is abstract;
11    procedure Imprime_Area (Item : in Object'Class);
12
13    type Circulo is new Object with record
14      Raio : Float;
15    end record;
16    overriding function Area (C : in Circulo) return Float;
17
18    type Retangulo is new Object with record
19      Largura, Altura : Float;
20    end record;
21    overriding function Area (R : in Retangulo) return Float;
22
23  end Geometria;
24
25  -----
26  -- Recete Geometria (Corpo)
27  package body Geometria is
28
29    overriding function Area (C : in Circulo) return Float is
30    begin
31      return Ada.Numerics.Pi * (C.Raio ** 2);
32    end Area;
33
34    overriding function Area (R : in Retangulo) return Float is
35    begin
36      return R.Largura * R.Altura;
37    end Area;
38
39  end Geometria;
40
41  -- AQUI ESTÁ A CORRÇÃO FINAL --
42  procedure Imprime_Area (Item : in Object'Class) is
43    begin
44      Ada.Text_IO.Put_Line ("A área do " & Item'Image);
45    end;

```

Output:

```

--- Testando o procedimento polimórfico ---
A área do Circulo é: 12.57
A área do Retangulo é: 20.00

```

- 7) Em um mundo tradicional mulheres não tem barbas e homens não amamentam os filhos. Porém, todas as pessoas têm uma data de nascimento. Declare um tipo **Person** com o componente comum **Birth** do tipo **Date** (conforme mostrado abaixo) e então derive os tipos **Man** e **Woman** que tenham componentes adicionais indicando se eles possuem barba ou não e quantos filhos eles amamentam respectivamente.

```

type Month_Name is (Jan, Feb, Mar, Apr, May, Jun,
Jul, Aug, Sep, Oct, Nov, Dec);

type Date is

```

```

record
  Day: Integer range 1..31;
  Month: Month_Name;
  Year: Integer;
end record;

```

```

-- Programa para Questões 7 e 8
with Ada.Text_IO; use Ada.Text_IO;
with Ada.Integer_Text_IO; -- Importação

procedure Solucao_7_e_8 is

  -- Definição do Pacote (Especificação)
  package People is

    -- Tipos base
    type Month_Name is (Jan, Feb, Mar, Apr, May, Jun,
                        Jul, Aug, Sep, Oct, Nov, Dec);

    type Date is record
      Day : Integer range 1..31;
      Month : Month_Name;
      Year : Integer;
    end record;

    -- Tipo Person e suas operações primitivas
    type Person is tagged record
      Birth : Date;
    end record;

    -- A operação de Person DEVE ser declarada ANTES dos filhos
    procedure Print_Details (P : in Person);

    -- Tipo Man e suas operações
    type Man is new Person with record
      Has_Beard : Boolean := False;
    end record;

    -- 'overriding'
    overriding
    procedure Print_Details (M : in Man);

    -- Tipo Woman e suas operações
    type Woman is new Person with record
      Children_Breastfed : Natural := 0;
    end record;

    overriding
    procedure Print_Details (W : in Woman);

    -- Procedimento polimórfico (pode ficar no final)
    procedure Analyze_Person (P : in Person'Class);

  end People;

  -- Corpo do Pacote
  package body People is

    package Month_IO is new Ada.Text_IO.Enumeration_IO (Month_Name);

```

```

-- Implementação de Print_Details para Person
procedure Print_Details (P : in Person) is
begin
  Put(" - [Person] Nascido em: ");
  -- Chamada para Integer_Text_IO
  Ada.Integer_Text_IO.Put(P.Birth.Day, Width => 2); Put("/");
  Month_IO.Put(P.Birth.Month); Put("/");
  Ada.Integer_Text_IO.Put(P.Birth.Year, Width => 4);
  New_Line;
end Print_Details;

-- Implementação de Print_Details para Man (sobrescrita)
overriding
procedure Print_Details (M : in Man) is
begin
  Print_Details(Person(M)); -- Chama o "pai"
  Put(" - [Man] ");
  if M.Has_Beard then
    Put_Line("Possui barba.");
  else
    Put_Line("Não possui barba.");
  end if;
end Print_Details;

-- Implementação de Print_Details para Woman (sobrescrita)
overriding
procedure Print_Details (W : in Woman) is
begin
  Print_Details(Person(W)); -- Chama o "pai"
  Put(" - [Woman] Amamentou ");
  -- Chamada para Integer_Text_IO
  Ada.Integer_Text_IO.Put(W.Children_Breastfed, Width => 1);
  Put_Line("filhos.");
end Print_Details;

-- Implementação de Analyze_Person
procedure Analyze_Person (P : in Person'Class) is
begin
  Put_Line("Analizando Objeto... ");
  Print_Details(P); -- Agora funciona!
end Analyze_Person;

end People;

-- Programa Principal (para testar)

use People;

-- Criando objetos de teste
Some_Person : Person := (Birth => (1, Jan, 1990));
Some_Man : Man := (Birth => (10, Mar, 1985), Has_Beard => True);
Some_Woman : Woman := (Birth => (20, Jul, 1992), Children_Breastfed => 2);

begin
  Put_Line("--- Teste Questões 7 & 8 (Polimorfismo ---");

  Analyze_Person(Some_Person);
  New_Line;

  Analyze_Person(Some_Man);
  New_Line;

  Analyze_Person(Some_Woman);
  New_Line;

end Solucao_7_e_8;

```

## IMAGEM DA EXECUÇÃO

```

HelloWorld.adb
Questões 7 e 8 (Polimorfismo)

1 -- Programa para Questões 7 e 8 (CORRIGIDO)
2 with Ada.Text_IO; use Ada.Text_IO;
3 with Ada.Integer_Text_IO; -- Importação correta
4
5 procedure Solucao_7_e_8 is
6
7   -- Definição do Pacote (Especificação) - ORDEM CORRIGIDA
8   package People is
9
10    -- Tipos base
11    type Month_Name is (Jan, Feb, Mar, Apr, May, Jun,
12                         Jul, Aug, Sep, Oct, Nov, Dec);
13
14    type Date is record
15      Day : Integer range 1..31;
16      Month : Month_Name;
17      Year : Integer;
18    end record;
19
20    -- Tipo Person e suas operações primitivas
21    type Person is tagged record
22      Birth : Date;
23    end record;
24
25    -- A operação de Person DEVE ser declarada ANTES dos filhos
26    procedure Print_Details (P : in Person);
27
28    -- Tipo Man e suas operações
29    type Man is new Person with record
30      Married : Boolean := False;
31    end record;
32
33    -- A operação 'overriding' está correta
34    overriding
35    procedure Print_Details (M : in Man);
36
37    -- Tipo Woman e suas operações
38    type Woman is new Person with record
39      Children_Breasted : Natural := 0;
40    end record;
41
42    -- 'overriding' aqui também está correto
43    overriding
44    procedure Print_Details (W : in Woman);
45
46

```

STDIN  
Input for the program (Optional)

Output:  
--- Teste Questões 7 & 8 (Polimorfismo) ---  
Analizando Objeto...  
- [Person] Nascido em: 1/JAN/1990  
Analizando Objeto...  
- [Person] Nascido em: 10/MAR/1985  
- [Man] Possui barba.  
Analizando Objeto...  
- [Person] Nascido em: 20/JUL/1992  
- [Woman] Amamentou 2 filhos.

- 8) Declare procedimentos *Print\_Details* para *Person*, *Man* e *Woman* que fornecem informações a respeito dos valores atuais dos componentes deles. Então declare um procedimento *Analyze\_Person* que recebe um parâmetro do tipo de uma classe ampla *Person'Class* e chama o procedimento apropriado *Print\_Details*.

```

-- Programa para Questões 7 e 8
with Ada.Text_IO; use Ada.Text_IO;
with Ada.Integer_Text_IO; -- Importação correta

procedure Solucao_7_e_8 is

  -- Definição do Pacote (Especificação)
  package People is

    -- Tipos base
    type Month_Name is (Jan, Feb, Mar, Apr, May, Jun,
                         Jul, Aug, Sep, Oct, Nov, Dec);

    type Date is record
      Day : Integer range 1..31;
      Month : Month_Name;
      Year : Integer;
    end record;

    -- Tipo Person e suas operações primitivas
    type Person is tagged record
      Birth : Date;
    end record;

    -- A operação 'overriding' está correta
    overriding
    procedure Print_Details (P : in Person);

    -- Tipo Man e suas operações
    type Man is new Person with record
      Married : Boolean := False;
    end record;

    -- A operação 'overriding' aqui também está correto
    overriding
    procedure Print_Details (M : in Man);

    -- Tipo Woman e suas operações
    type Woman is new Person with record
      Children_Breasted : Natural := 0;
    end record;
  end;

```

```

end record;

-- A operação de Person DEVE ser declarada ANTES dos filhos
procedure Print_Details (P : in Person);

-- Tipo Man e suas operações
type Man is new Person with record
  Has_Beard : Boolean := False;
end record;

-- 'overriding'
overriding
procedure Print_Details (M : in Man);

-- Tipo Woman e suas operações
type Woman is new Person with record
  Children_Breastfed : Natural := 0;
end record;

overriding
procedure Print_Details (W : in Woman);

-- Procedimento polimórfico (pode ficar no final)
procedure Analyze_Person (P : in Person'Class);

end People;

-- Corpo do Pacote
package body People is

  package Month_IO is new Ada.Text_IO.Enumeration_IO (Month_Name);

  -- Implementação de Print_Details para Person
  procedure Print_Details (P : in Person) is
    begin
      Put(" - [Person] Nascido em: ");
      -- Chamada para Integer_Text_IO
      Ada.Integer_Text_IO.Put(P.Birth.Day, Width => 2); Put("/");
      Month_IO.Put(P.Birth.Month); Put("/");
      Ada.Integer_Text_IO.Put(P.Birth.Year, Width => 4);
      New_Line;
    end Print_Details;

  -- Implementação de Print_Details para Man (sobrescrita)
  overriding
  procedure Print_Details (M : in Man) is
    begin
      Print_Details(Person(M)); -- Chama o "pai"
      Put(" - [Man] ");
      if M.Has_Beard then
        Put_Line("Possui barba.");
      else
        Put_Line("Não possui barba.");
      end if;
    end Print_Details;

  -- Implementação de Print_Details para Woman (sobrescrita)
  overriding
  procedure Print_Details (W : in Woman) is
    begin
      Print_Details(Person(W)); -- Chama o "pai"
      Put(" - [Woman] Amamentou ");
      -- Chamada para Integer_Text_IO
      Ada.Integer_Text_IO.Put(W.Children_Breastfed, Width => 1);
      Put_Line(" filhos.");
    end Print_Details;

```

```

end Print_Details;

-- Implementação de Analyze_Person
procedure Analyze_Person (P : in Person'Class) is
begin
    Put_Line("Analizando Objeto...");
    Print_Details(P); -- Agora funciona!
end Analyze_Person;

end People;

-- Programa Principal (para testar)

use People;

-- Criando objetos de teste
Some_Person : Person := (Birth => (1, Jan, 1990));
Some_Man   : Man   := (Birth => (10, Mar, 1985), Has_Beard => True);
Some_Woman  : Woman := (Birth => (20, Jul, 1992), Children_Breastfed => 2);

begin
    Put_Line("--- Teste Questões 7 & 8 (Polimorfismo ---");

    Analyze_Person(Some_Person);
    New_Line;

    Analyze_Person(Some_Man);
    New_Line;

    Analyze_Person(Some_Woman);
    New_Line;

end Solucao_7_e_8;

```

## IMAGEM DA EXECUÇÃO

The screenshot shows the OneCompiler IDE interface. The code editor displays the Ada code for 'Solucao\_7\_e\_8'. The output window shows the execution results:

```

STDIN
Input for the program (Optional)

Output:
--- Teste Questões 7 & 8 (Polimorfismo ---
Analizando Objeto...
- [Person] Nascido em: 1/JAN/1990
Analizando Objeto...
- [Person] Nascido em: 10/MAR/1985
- [Man] Possui barba.
Analizando Objeto...
- [Person] Nascido em: 20/JUL/1992
- [Woman] Amamentou 2 filhos.

```

9) Escreva a especificação e o corpo do pacote *Queues* usando uma implementação de lista encadeada.

```
-- Programa para Questão 9
with Ada.Text_IO; use Ada.Text_IO;
with Ada.Integer_Text_IO;
with Ada.Unchecked_Deallocation;

procedure Solucao_9 is

    -- Especificação do Pacote
    package Integer_Queue is

        type Queue is limited private;
        Queue_Error : exception;

        procedure Enqueue (Q : in out Queue; Item : in Integer);
        function Dequeue (Q : in out Queue) return Integer;
        function Front (Q : in Queue) return Integer;
        function Is_Error (Q : in Queue) return Boolean;

    private
        type Node;
        type Node_Ptr is access Node;
        type Node is record
            Element : Integer;
            Next    : Node_Ptr;
        end record;

        type Queue is limited record
            Head : Node_Ptr := null;
            Tail : Node_Ptr := null;
        end record;

    end Integer_Queue;

    -- Corpo do Pacote
    package body Integer_Queue is

        procedure Free is new Ada.Unchecked_Deallocation(Node, Node_Ptr);

        procedure Enqueue (Q : in out Queue; Item : in Integer) is
            New_Node : constant Node_Ptr := new Node'(Element => Item, Next => null);
        begin
            if Q.Tail = null then
                Q.Head := New_Node;
                Q.Tail := New_Node;
            else
                Q.Tail.Next := New_Node;
                Q.Tail := New_Node;
            end if;
        end Enqueue;

        function Dequeue (Q : in out Queue) return Integer is
            Old_Head : Node_Ptr;
            Item      : Integer;
        begin
            if Is_Error(Q) then
                raise Queue_Error;
            end if;
            Old_Head := Q.Head;
            Item := Old_Head.Element;
            Q.Head := Old_Head.Next;
            if Q.Head = null then
                Q.Tail := null;
            end if;
            Free(Old_Head);
        end Dequeue;

        function Is_Error (Q : in Queue) return Boolean is
            Node_Ptr : Node_Ptr;
        begin
            if Q.Head = null then
                Node_Ptr := Q.Tail;
            else
                Node_Ptr := Q.Head;
            end if;
            if Node_Ptr = null then
                return True;
            else
                return False;
            end if;
        end Is_Error;

    end Integer_Queue;
```

```

        raise Queue_Error;
    end if;

    Old_Head := Q.Head;
    Item := Old_Head.Element;
    Q.Head := Old_Head.Next;

    if Q.Head = null then
        Q.Tail := null;
    end if;

    Free(Old_Head);
    return Item;
end Dequeue;

function Front (Q : in Queue) return Integer is
begin
    if Is_Error(Q) then
        raise Queue_Error;
    end if;
    return Q.Head.Element;
end Front;

function Is_Error (Q : in Queue) return Boolean is
begin
    return Q.Head = null;
end Is_Error;

end Integer_Queues;

-- Programa Principal (para testar no OneCompiler)
use Integer_Queues;
My_Int_Queue : Queue;

begin
    Put_Line("--- Teste Questão 9 (Fila de Inteiros) ---");

    if Is_Error(My_Int_Queue) then Put_Line("Fila criada. Está vazia."); end if;

    Enqueue(My_Int_Queue, 10);
    Enqueue(My_Int_Queue, 20);
    Put_Line("Adicionados 10 e 20.");

    Put_Line("Elemento da frente: " & Integer'Image(Front(My_Int_Queue)));
    Put_Line("Removendo: " & Integer'Image(Dequeue(My_Int_Queue)));
    Put_Line("Elemento da frente: " & Integer'Image(Front(My_Int_Queue)));
    Put_Line("Removendo: " & Integer'Image(Dequeue(My_Int_Queue)));

    if Is_Error(My_Int_Queue) then Put_Line("Fila está vazia novamente."); end if;

exception
    when Queue_Error =>
        Put_Line("ERRO: Tentou remover de fila vazia!");
end Solucao_9;

```

IMAGEM DA EXECUÇÃO

The screenshot shows the OneCompiler IDE interface. In the top navigation bar, there are buttons for AI, NEW, Pricing, Learn, Code, Deploy, and More. Below the navigation bar, the code editor window is titled "HelloWorld.adb". The code itself is an Ada program named "Solucao\_9" which implements a queue. It includes declarations for packages, types (like Queue, Node, and Queue\_Ptr), and procedures (Enqueue, Dequeue, Front, Is\_Empty). The package body contains the implementation of these procedures, including memory allocation and deallocation using Ada.Unchecked\_Deallocation. The output window shows the results of a test run where a queue is created, elements are added (10 and 20), and then removed (20), resulting in an empty queue again.

```

1  -- Programa para Questão 9
2  with Ada.Text_IO; use Ada.Text_IO;
3  with Ada.Integer_Text_IO;
4  with Ada.Unchecked_Deallocation;
5
6  procedure Solucao_9 is
7
8      -- Especificação do Pacote
9      package Integer_Queue is
10
11          type Queue is limited private;
12          Queue_Error : exception;
13
14          procedure Enqueue (Q : in out Queue; Item : in Integer);
15          function Dequeue (Q : in out Queue) return Integer;
16          function Front (Q : in Queue) return Integer;
17          function Is_Error (Q : in Queue) return Boolean;
18
19      private
20          type Node;
21          type Node_Ptr is access Node;
22          type Node is record
23              Element : Integer;
24              Next   : Node_Ptr;
25          end record;
26
27          type Queue is limited record
28              Head : Node_Ptr := null;
29              Tail : Node_Ptr := null;
30          end record;
31
32      end Integer_Queue;
33
34      -- Corpo do Pacote
35      package body Integer_Queue is
36
37          procedure Free is new Ada.Unchecked_Deallocation(Node, Node_Ptr);
38
39          procedure Enqueue (Q : in out Queue; Item : in Integer) is
40              New_Node : constant Node_Ptr := new Node'(Element => Item, Next => null);
41          begin
42              if Q.Tail = null then
43                  Q.Head := New_Node;
44                  Q.Tail := New_Node;
45              else

```

Output:

```

-- Teste Questão 9 (Fila de Inteiros) --
Fila criada. Está vazia.
Adicionando 10 e 20.
Elemento da frente: 10
Removendo: 10
Elemento da frente: 20
Removendo: 20
Fila está vazia novamente.

```

10) Escreva um pacote *generic* para *Queues* tal que filas de qualquer tipo possam ser declaradas.

```

-- Programa para Questão 10
with Ada.Text_IO; use Ada.Text_IO;
with Ada.Unchecked_Deallocation;
with Ada.Strings.Unbounded;
use Ada.Strings.Unbounded;

procedure Solucao_10 is

    -- Especificação do Pacote Genérico
    generic
        type Element_Type is private;
        package Generic_Queue is

            type Queue is limited private;
            Queue_Error : exception;

            procedure Enqueue (Q : in out Queue; Item : in Element_Type);
            function Dequeue (Q : in out Queue) return Element_Type;
            function Front (Q : in Queue) return Element_Type;
            function Is_Error (Q : in Queue) return Boolean;

        private
            type Node;
            type Node_Ptr is access Node;
            type Node is record
                Element : Element_Type;
                Next   : Node_Ptr;
            end record;

            type Queue is limited record
                Head : Node_Ptr := null;
                Tail : Node_Ptr := null;
            end record;

        end Generic_Queue;

```

```

-- Corpo do Pacote Genérico

package body Generic_Queue is

procedure Free is new Ada.Unchecked_Deallocation(Node, Node_Ptr);

procedure Enqueue (Q : in out Queue; Item : in Element_Type) is
    New_Node : constant Node_Ptr := new Node'(Element => Item, Next => null);
begin
    if Q.Tail = null then
        Q.Head := New_Node;
        Q.Tail := New_Node;
    else
        Q.Tail.Next := New_Node;
        Q.Tail := New_Node;
    end if;
end Enqueue;

function Dequeue (Q : in out Queue) return Element_Type is
    Old_Head : Node_Ptr;
    Item    : Element_Type;
begin
    if Is_Empty(Q) then
        raise Queue_Error;
    end if;

    Old_Head := Q.Head;
    Item := Old_Head.Element;
    Q.Head := Old_Head.Next;

    if Q.Head = null then
        Q.Tail := null;
    end if;

    Free(Old_Head);
    return Item;
end Dequeue;

function Front (Q : in Queue) return Element_Type is
begin
    if Is_Empty(Q) then
        raise Queue_Error;
    end if;
    return Q.Head.Element;
end Front;

function Is_Empty (Q : in Queue) return Boolean is
begin
    return Q.Head = null;
end Is_Empty;

end Generic_Queue;

-- Programa Principal (para testar no OneCompiler)

package String_Queue is new Generic_Queue(Element_Type => Unbounded_String);
use String_Queue;

My_String_Queue : Queue;

begin
    Put_Line("--- Teste Questão 10 (Fila Genérica de Strings) ---");

    Enqueue(My_String_Queue, To_Unbounded_String("Ola"));

```

```

Enqueue(My_String_Queue, To_Unbounded_String("Mundo"));
Put_Line("Adicionados 'Ola' e 'Mundo'.");

Put_Line("Elemento da frente: " & To_String(Front(My_String_Queue)));
Put_Line("Removendo: " & To_String(Dequeue(My_String_Queue)));
Put_Line("Removendo: " & To_String(Dequeue(My_String_Queue)));

if Is_Empty(My_String_Queue) then Put_Line("Fila de strings vazia."); end if;

exception
when Queue_Error =>
    Put_Line("ERRO: Tentou remover de fila vazia!");

end Solucao_10;

```

## IMAGEM DA EXECUÇÃO

The screenshot shows the OneCompiler IDE interface. In the top bar, there are tabs for AI, NEW, ADA, and RUN. The ADA tab is selected. Below the tabs, there is a search bar and a menu bar with options like Pricing, Learn, Code, Deploy, and More.

In the main area, there is an input field labeled "Input for the program (Optional)" containing the string "... Teste Questão 10 (Fila Genérica de Strings) ...".

The output window displays the following text:

```

...
Adicionados 'Ola' e 'Mundo'.
Elemento da frente: Ola
Removendo: Ola
Removendo: Mundo
Fila de strings vazia.

```

- 11) Escreva a especificação e o corpo do pacote **Stacks** usando uma implementação de lista encadeada.

```

with Ada.Text_IO; use Ada.Text_IO;
with Ada.Integer_Text_IO;
with Ada.Unchecked_Deallocation;

```

```

procedure Solucao_11 is

-----
-- QUESTÃO 11: Pacote Stacks (Pilha de Inteiros)
-----
package Integer_Stacks is

    type Stack is limited private;
    Stack_Error : exception;

    procedure Push (S : in out Stack; Item : in Integer);
    function Pop (S : in out Stack) return Integer;
    function Top (S : in Stack) return Integer;
    function Is_Error (S : in Stack) return Boolean;

private
    type Node;
    type Node_Ptr is access Node;
    type Node is record
        Element : Integer;
        Next   : Node_Ptr;
    end record;

    type Stack is limited record
        Top_Node : Node_Ptr := null;
    end record;

end Integer_Stacks;

-- Corpo do pacote Integer_Stacks
package body Integer_Stacks is

    procedure Free is new Ada.Unchecked_Deallocation(Node, Node_Ptr);

    procedure Push (S : in out Stack; Item : in Integer) is
        -- O novo nó aponta para o topo antigo
        New_Node : constant Node_Ptr := new Node'(Element => Item, Next => S.Top_Node);
    begin
        -- O topo agora é o novo nó
        S.Top_Node := New_Node;
    end Push;

    function Pop (S : in out Stack) return Integer is
        Old_Top : Node_Ptr;
        Item   : Integer;
    begin
        if Is_Error(S) then
            raise Stack_Error;
        end if;

        Old_Top := S.Top_Node;
        Item := Old_Top.Element;

        S.Top_Node := Old_Top.Next;
        Free(Old_Top);
        return Item;
    end Pop;

    function Top (S : in Stack) return Integer is
    begin
        if Is_Error(S) then
            raise Stack_Error;
        end if;
        return S.Top_Node.Element;
    end Top;

```

```

function Is_Empty (S : in Stack) return Boolean is
begin
    return S.Top_Node = null;
end Is_Empty;

end Integer_Stacks;

-- Bloco principal para testar a Solução 11
use Integer_Stacks;
My_Int_Stack : Stack;

begin
    Put_Line("--- Teste Questão 11 (Pilha de Inteiros) ---");

    if Is_Empty(My_Int_Stack) then Put_Line("Pilha criada. Está vazia."); end if;

    Push(My_Int_Stack, 100);
    Push(My_Int_Stack, 200);
    Put_Line("Adicionados 100 e 200");

    Put_Line("Elemento do topo: " & Integer'Image(Top(My_Int_Stack)));
    Put_Line("Removendo: " & Integer'Image(Pop(My_Int_Stack)));
    Put_Line("Elemento do topo: " & Integer'Image(Top(My_Int_Stack)));
    Put_Line("Removendo: " & Integer'Image(Pop(My_Int_Stack)));

    if Is_Empty(My_Int_Stack) then Put_Line("Pilha está vazia novamente."); end if;

exception
    when Stack_Error =>
        Put_Line("ERRO: Tentou remover de pilha vazia!");

end Solucao_11;

```

## IMAGEM DA EXECUÇÃO

The screenshot shows the OneCompiler IDE interface. The code in the editor is identical to the one above. In the output window, the following text is displayed:

```

--- Teste Questão 11 (Pilha de Inteiros) ---
Pilha criada. Esta vazia.
Adicionados 100 e 200.
Elemento do topo: 200
Removendo: 200
Elemento do topo: 100
Removendo: 100
Pilha está vazia novamente.

```

- 12) Escreva um pacote **generic** para **Stacks** tal que pilhas de qualquer tipo possam ser declaradas.

```

-- Programa para Questão 12

with Ada.Text_IO; use Ada.Text_IO;
with Ada.Float_Text_IO; -- Importa o pacote de I/O para Floats
with Ada.Unchecked_Deallocation;

procedure Solucao_12 is

-----  

-- QUESTÃO 12: Pacote Genérico para Stacks  

-----  

generic
    type Element_Type is private;
    package Generic_Stacks is

        type Stack is limited private;
        Stack_Error : exception;

        procedure Push (S : in out Stack; Item : in Element_Type);
        function Pop (S : in out Stack) return Element_Type;
        function Top (S : in Stack) return Element_Type;
        function Is_Error (S : in Stack) return Boolean;

    private
        type Node;
        type Node_Ptr is access Node;
        type Node is record
            Element : Element_Type;
            Next   : Node_Ptr;
        end record;

        type Stack is limited record
            Top_Node : Node_Ptr := null;
        end record;

    end Generic_Stacks;

-- Corpo do pacote Generic_Stacks

package body Generic_Stacks is

    procedure Free is new Ada.Unchecked_Deallocation(Node, Node_Ptr);

    procedure Push (S : in out Stack; Item : in Element_Type) is
        New_Node : constant Node_Ptr := new Node'(Element => Item, Next => S.Top_Node);
    begin
        S.Top_Node := New_Node;
        end Push;

    function Pop (S : in out Stack) return Element_Type is
        Old_Top : Node_Ptr;
        Item    : Element_Type;
    begin
        if Is_Error(S) then
            raise Stack_Error;
        end if;

        Old_Top := S.Top_Node;
        Item := Old_Top.Element;

        S.Top_Node := Old_Top.Next;
        Free(Old_Top);
        return Item;
    end Pop;

    function Top (S : in Stack) return Element_Type is
    begin

```

```

if Is_Empty(S) then
    raise Stack_Empty;
end if;
return S.Top_Node.Element;
end Top;

function Is_Empty (S : in Stack) return Boolean is
begin
    return S.Top_Node = null;
end Is_Empty;

end Generic_Stacks;

-- Bloco principal para testar a Solução 12

-- Instanciando o pacote genérico para o tipo Float
package Float_Stacks is new Generic_Stacks(Element_Type => Float);
use Float_Stacks;

-- Não há necessidade de instanciar o Ada.Float_Text_IO

My_Float_Stack : Stack;

begin
Put_Line("--- Teste Questão 12 (Pilha Genérica de Floats) ---");

if Is_Empty(My_Float_Stack) then Put_Line("Pilha criada. Está vazia."); end if;

Push(My_Float_Stack, 3.14);
Push(My_Float_Stack, 1.618);
Put_Line("Adicionados 3.14 e 1.618.");

-- Chamando 'Ada.Float_Text_IO.Put'
-- e adicionando formatação (Fore, Aft, Exp).
Put("Elemento do topo: ");
Ada.Float_Text_IO.Put(Top(My_Float_Stack), Fore => 1, Aft => 3, Exp => 0);
New_Line;

Put("Removendo: ");
Ada.Float_Text_IO.Put(Pop(My_Float_Stack), Fore => 1, Aft => 3, Exp => 0);
New_Line;

Put("Elemento do topo: ");
Ada.Float_Text_IO.Put(Top(My_Float_Stack), Fore => 1, Aft => 2, Exp => 0);
New_Line;

Put("Removendo: ");
Ada.Float_Text_IO.Put(Pop(My_Float_Stack), Fore => 1, Aft => 2, Exp => 0);
New_Line;

if Is_Empty(My_Float_Stack) then Put_Line("Pilha de floats vazia."); end if;

exception
when Stack_Error =>
    Put_Line("ERRO: Tentou remover de pilha vazia!");

end Solucao_12;

```

## IMAGEM DA EXECUÇÃO

The screenshot shows the OneCompiler IDE interface. On the left is the code editor with Ada code for Question 12. On the right is the output window showing the execution results.

```

-- Bloco principal para testar a Solução 12
-- Instanciando o pacote genérico para o tipo Float
package Float_Stacks is new Generic_Stacks(Element_Type => Float);
use Float_Stacks;
-- Não há necessidade de instanciar o Ada.Float_Text_IO
My_Float_Stack : Stack;
begin
  Put_Line("... Teste Questão 12 (Pilha Genérica de Floats) ...");
  if Is_Empty(My_Float_Stack) then Put_Line("Pilha criada. Está vazia."); end if;
  Push(My_Float_Stack, 3.14);
  Push(My_Float_Stack, 1.618);
  Put_Line("Adicionados 3.14 e 1.618.");
  -- Chamando 'Ada.Float_Text_IO.Put'
  Put("Elemento do topo: ");
  Ada.Float_Text_IO.Put(Top(My_Float_Stack), Fore => 1, Aft => 3, Exp => 0);
  New_Line;
  Put("Removendo ...");
  Ada.Float_Text_IO.Put(Pop(My_Float_Stack), Fore => 1, Aft => 3, Exp => 0);
  New_Line;
  Put("Elemento do topo: ");
  Ada.Float_Text_IO.Put(Top(My_Float_Stack), Fore => 1, Aft => 3, Exp => 0);
  New_Line;
  Put("Removendo ...");
  Ada.Float_Text_IO.Put(Pop(My_Float_Stack), Fore => 1, Aft => 3, Exp => 0);
  New_Line;
  if Is_Empty(My_Float_Stack) then Put_Line("Pilha de floats vazia."); end if;
exception
  when Stack_Error =>
    Put_Line("ERRO: Tentou remover de pilha vazia!");
end;

```

Output:

```

--- Teste Questão 12 (Pilha Genérica de Floats) ---
Pilha criada. Está vazia.
Adicionados 3.14 e 1.618.
Elemento do topo: 3.14
Removendo: 3.14
Elemento do topo: 1.618
Removendo: 1.618
Elemento do topo: 3.14
Removendo: 3.14
Pilha de floats vazia.

```

13) Crie um tipo de dado abstrato (TDA) chamado Retângulo. O TDA tem atributos comprimento e largura cada um com valor default igual a 1. Ele tem funções “membro” que calculam o *comprimento*, *largura*, *perímetro* e *área* do retângulo. Forneça as funções *set* e *get*, tanto para o comprimento como para a largura. A função *set* deve verificar se o comprimento e a largura são números de ponto flutuante maiores que 0.0 e menores que 20.0.

Além disso, a função *set* especifica se as coordenadas fornecidas de fato especificam um retângulo. Lembre que o comprimento é a maior das duas dimensões. Inclua uma função quadrado, que determina se um retângulo é um quadrado.

```

-- Programa para Questão 13: TDA Retângulo
-- Saída de "É um quadrado" / "Não é um quadrado"

with Ada.Text_IO;      use Ada.Text_IO;
with Ada.Float_Text_IO; use Ada.Float_Text_IO;
with Ada.Exceptions;   use Ada.Exceptions;

procedure Solucao_13 is

-----  

-- ESPECIFICAÇÃO DO TDA RETÂNGULO  

-----  

package Retangulos is
  type Retangulo is private;

  function Get_Comprimento(R : Retangulo) return Float;
  function Get_Largura(R : Retangulo) return Float;
  procedure Set_Dimensoes(R : in out Retangulo; C, L : Float);
  function Perimetro(R : Retangulo) return Float;
  function Area(R : Retangulo) return Float;
  function Quadrado(R : Retangulo) return Boolean;

private

```

```

type Retangulo is
  record
    Comprimento : Float := 1.0;
    Largura    : Float := 1.0;
  end record;
end Retangulos;

-----
-- CORPO (IMPLEMENTAÇÃO) DO TDA RETÂNGULO
-----

package body Retangulos is

  function Get_Comprimento(R : Retangulo) return Float is
  begin
    return R.Comprimento;
  end Get_Comprimento;

  function Get_Largura(R : Retangulo) return Float is
  begin
    return R.Largura;
  end Get_Largura;

  procedure Set_Dimensoes(R : in out Retangulo; C, L : Float) is
  begin
    -- Verifica se C e L são maiores que 0.0 e menores que 20.0
    if C > 0.0 and C < 20.0 and L > 0.0 and L < 20.0 then

      -- Garante que o Comprimento seja a maior dimensão
      if C >= L then
        R.Comprimento := C;
        R.Largura    := L;
      else
        R.Comprimento := L;
        R.Largura    := C;
      end if;
    else
      -- Se a validação falhar, lança uma exceção.
      raise Constraint_Error;
    end if;
  end Set_Dimensoes;

  function Perimetro(R : Retangulo) return Float is
  begin
    return 2.0 * (R.Comprimento + R.Largura);
  end Perimetro;

  function Area(R : Retangulo) return Float is
  begin
    return R.Comprimento * R.Largura;
  end Area;

  function Quadrado(R : Retangulo) return Boolean is
  begin
    -- Um quadrado é quando o comprimento e a largura são iguais.
    return R.Comprimento = R.Largura;
  end Quadrado;

end Retangulos;

-----
-- PROGRAMA PRINCIPAL PARA TESTAR O TDA
-----
use Retangulos; -- Torna 'Retangulo', 'Set_Dimensoes', etc. visíveis

-- Procedimento auxiliar para imprimir todos os detalhes
procedure Imprimir_Detalhes(Nome: String; R: Retangulo) is

```

```

begin
  New_Line;
  Put_Line("--- Detalhes de: " & Nome & " ---");
  Put("Comprimento: "); Put(Get_Comprimento(R), Fore => 1, Aft => 2, Exp => 0); New_Line;
  Put("Largura:   "); Put(Get_Largura(R),   Fore => 1, Aft => 2, Exp => 0); New_Line;
  Put("Perimetro: "); Put(Perimetro(R),   Fore => 1, Aft => 2, Exp => 0); New_Line;
  Put("Area:      "); Put(Area(R),      Fore => 1, Aft => 2, Exp => 0); New_Line;

  if Quadrado(R) then
    Put_Line("Resultado: É um quadrado.");
  else
    Put_Line("Resultado: Não é um quadrado.");
  end if;
end Imprimir_Detalhes;

-- Início do programa de teste
begin

  -- Teste 1: Retângulo Default (deve ser um quadrado)
  declare
    R1 : Retangulo; -- Usa os valores default (1.0, 1.0)
  begin
    Imprimir_Detalhes("R1 (Default)", R1);
  end;

  -- Teste 2: Retângulo Válido (não deve ser um quadrado)
  declare
    R2 : Retangulo;
  begin
    Set_Dimensoes(R2, 10.0, 10.5); -- TDA armazena 10.0 e 7.5
    Imprimir_Detalhes("R2 (10.0, 10.5)", R2);
  end;

  -- Teste 3: Tentativa de valor inválido
  declare
    R3 : Retangulo;
  begin
    New_Line;
    Put_Line("--- Teste 3: Definindo R3 (Invalido: 25.0) ---");
    begin
      Set_Dimensoes(R3, 25.0, 10.0);
      Put_Line("ERRO: O programa deveria ter parado!");
    exception
      when E: Constraint_Error =>
        Put_Line("OK! Execção 'Constraint_Error' capturada como esperado.");
        Put_Line("Os valores do retângulo não foram alterados.");
        Imprimir_Detalhes("R3 (Valores default mantidos)", R3);
    end;
  end;

end Solucao_13;

```

## IMAGEM DA EXECUÇÃO

```

1 -- Programa para Questão 13: TDA Retângulo
2 -- Saída de "é um quadrado" / "Não é um quadrado"
3
4 with Ada.Text_IO;      use Ada.Text_IO;
5 with Ada.Float_Text_IO; use Ada.Float_Text_IO;
6 with Ada.Exceptions;   use Ada.Exceptions;
7
8 procedure Solucao_13 is
9
10    -- ESPECIFICAÇÃO DO TDA RETÂNGULO
11
12    package Retangulos is
13        type Retangulo is private;
14
15        function Get_Comprimento(R : Retangulo) return Float;
16        function Get_Largura(R : Retangulo) return Float;
17        procedure Set_Dimensao(C, L : in out Retangulo; C, L : Float);
18        function Perimetro(R : Retangulo) return Float;
19        function Area(R : Retangulo) return Float;
20        function Quadrado(R : Retangulo) return Boolean;
21
22    private
23        type Retangulo is
24            record
25                Comprimento : Float := 1.0;
26                Largura     : Float := 1.0;
27            end record;
28    end Retangulos;
29
30
31    -- CORPO (IMPLEMENTAÇÃO) DO TDA RETÂNGULO
32
33    package body Retangulos is
34
35        function Get_Comprimento(R : Retangulo) return Float is
36        begin
37            return R.Comprimento;
38        end Get_Comprimento;
39
40        function Get_Largura(R : Retangulo) return Float is
41        begin
42            return R.Largura;
43        end Get_Largura;
44
45    end Retangulos;

```

Output:

```

--- Detalhes de: R1 (Default) ---
Comprimento: 1.00
Largura: 1.00
Perímetro: 4.00
Área: 1.00
Resultado: É um quadrado.

--- Detalhes de: R2 (10.0, 10.5) ---
Comprimento: 10.50
Largura: 10.00
Perímetro: 41.00
Área: 105.00
Resultado: Não é um quadrado.

--- Teste 3: Definindo R3 (Invalido: 25.0) ---
OK! Exceção 'Constraint_Error' capturada como esperado.
Os valores do retângulo não foram alterados.

--- Detalhes de: R3 (Valores default mantidos) ---
Comprimento: 1.00
Largura: 1.00
Perímetro: 4.00
Área: 1.00
Resultado: É um quadrado.

```

14) Crie um TDA Racional para fazer aritmética com frações. Escreva um programa para testar seu TDA. Use variáveis inteiros para representar os dados **private** do TDA – o numerador e o denominador. Forneça uma “função” construtor que permita que um objeto deste TDA seja inicializado quando é declarado. O construtor deve conter valores *default* no caso de nenhum inicializador ser fornecido e deve armazenar a fração em formato reduzido. Forneça funções membro para cada um dos seguintes itens:

- a) Adição, subtração, multiplicação e divisão de dois números do tipo Racional.
- b) Imprimir números do tipo Racional no formato  $a/b$  alinhado a esquerda. c) Imprimir números do tipo Racional em formato de ponto flutuante com 3 dígitos de precisão.

```

-- Programa para Questão 14: TDA Racional
with Ada.Text_IO;      use Ada.Text_IO;
with Ada.Integer_Text_IO; use Ada.Integer_Text_IO;
with Ada.Float_Text_IO; use Ada.Float_Text_IO;
with Ada.Exceptions;   use Ada.Exceptions;

procedure Solucao_14 is

-----  

-- ESPECIFICAÇÃO DO TDA RACIONAL  

-----  

package Racional_TDA is

    type Racional is private;

    -- Função "Construtor":
    -- 1. Fornece valores default (0, 1) se nenhum for dado.
    -- 2. Garante o formato reduzido e o sinal correto.
    function Criar(N : Integer := 0; D : Integer := 1) return Racional;

```

```

-- Funções de "get" (úteis para as operações)
function Get_Num(R : Racional) return Integer;
function Get_Den(R : Racional) return Positive;

-- a) Funções de aritmética (sobrecarga de operadores)
function "+"(L, R : Racional) return Racional;
function "-"(L, R : Racional) return Racional;
function "*"(L, R : Racional) return Racional;
function "/"(L, R : Racional) return Racional;

-- b) Imprimir em formato fração (a / b)
procedure Imprimir_Fracao(R : in Racional);

-- c) Imprimir em formato ponto flutuante
procedure Imprimir_Float(R : in Racional);

private
  -- Parte privada da especificação
  type Racional is record
    -- O denominador é 'Positive' para garantir que nunca seja 0 ou negativo.
    Num : Integer := 0;
    Den : Positive := 1;
  end record;

end Racional_TDA;

-----
-- CORPO (IMPLEMENTAÇÃO) DO TDA RACIONAL
-----
package body Racional_TDA is

  -- Função auxiliar (privada) para calcular o Máximo Divisor Comum (MDC)
  -- Usada para reduzir as frações.
  function GCD(A, B : Integer) return Integer is
    Val_A : Integer := abs A;
    Val_B : Integer := abs B;
    Temp : Integer;
  begin
    if Val_A = 0 then return Val_B; end if;
    if Val_B = 0 then return Val_A; end if;

    while Val_B > 0 loop
      Temp := Val_A mod Val_B;
      Val_A := Val_B;
      Val_B := Temp;
    end loop;
    return Val_A;
  end GCD;

  -- Implementação do "Construtor"
  function Criar(N : Integer := 0; D : Integer := 1) return Racional is
    Temp_N : Integer := N;
    Temp_D : Integer := D;
    G : Integer;
  begin
    -- Validação do denominador
    if D = 0 then
      Put_Line("ERRO: Denominador não pode ser zero.");
      raise Constraint_Error;
    end if;

    -- Normalização do Sinal: O sinal fica sempre no numerador.
    if Temp_D < 0 then
      Temp_N := -Temp_N;
      Temp_D := -Temp_D;
    end if;
  end Criar;

```

```

-- Redução da Fração
G := GCD(Temp_N, Temp_D);

-- Retorna o novo objeto Racional reduzido
return (Num => Temp_N / G, Den => Positive(Temp_D / G));
end Criar;

-- Implementação das funções "get"
function Get_Num(R : Racional) return Integer is (R.Num);
function Get_Den(R : Racional) return Positive is (R.Den);

-- a) Implementação da Aritmética

function "+"(L, R : Racional) return Racional is
    Novo_Num : constant Integer := (L.Num * R.Den) + (R.Num * L.Den);
    Novo_Den : constant Integer := (L.Den * R.Den);
begin
    -- Retorna a fração já reduzida pelo "construtor"
    return Criar(Novo_Num, Novo_Den);
end "+";

function "-"(L, R : Racional) return Racional is
    Novo_Num : constant Integer := (L.Num * R.Den) - (R.Num * L.Den);
    Novo_Den : constant Integer := (L.Den * R.Den);
begin
    return Criar(Novo_Num, Novo_Den);
end "-";

function "*"(L, R : Racional) return Racional is
    Novo_Num : constant Integer := L.Num * R.Num;
    Novo_Den : constant Integer := L.Den * R.Den;
begin
    return Criar(Novo_Num, Novo_Den);
end "*";

function "/"(L, R : Racional) return Racional is
    -- Divisão é a multiplicação pelo inverso
    Novo_Num : constant Integer := L.Num * R.Den;
    Novo_Den : constant Integer := L.Den * R.Num;
begin
    return Criar(Novo_Num, Novo_Den);
end "/";

-- b) Implementação da Impressão (Fração)
procedure Imprimir_Fracao(R : in Racional) is
begin
    Put(Item => R.Num, Width => 1);
    Put("/");
    Put(Item => R.Den, Width => 1);
end Imprimir_Fracao;

-- c) Implementação da Impressão (Float)
procedure Imprimir_Float(R : in Racional) is
    Resultado : constant Float := Float(R.Num) / Float(R.Den);
begin
    -- Formata com 3 dígitos de precisão
    Put(Resultado, Fore => 1, Aft => 3, Exp => 0);
end Imprimir_Float;

end Racional_TDA;

-----
-- PROGRAMA PRINCIPAL PARA TESTAR O TDA
-----
use Racional_TDA; -- Torna 'Racional', 'Criar', '+', etc. visíveis

```

```

begin
  Put_Line("--- Teste TDA Racional ---");

  -- Teste 1: Declaração e Redução
  New_Line;
  Put_Line("Teste 1: Criacao e Reducao");
declare
  R1 : Racional := Criar(10, 20); -- Deve reduzir para 1 / 2
  R2 : Racional := Criar(7, -21); -- Deve reduzir para -1 / 3
begin
  Put("R1 (10/20) = "); Imprimir_Fracao(R1); New_Line;
  Put("R2 (7/-21) = "); Imprimir_Fracao(R2); New_Line;
end;

  -- Teste 2: Construtor Default
  New_Line;
  Put_Line("Teste 2: Construtor Default");
declare
  R_Default : Racional := Criar; -- Deve ser 0 / 1
begin
  Put("R_Default = "); Imprimir_Fracao(R_Default); New_Line;
end;

  -- Teste 3: Aritmética (a)
  New_Line;
  Put_Line("Teste 3: Aritmetica (1/2 e 1/3)");
declare
  A : Racional := Criar(1, 2);
  B : Racional := Criar(1, 3);
begin
  Put("A = "); Imprimir_Fracao(A); New_Line;
  Put("B = "); Imprimir_Fracao(B); New_Line;

  Put("A + B = "); Imprimir_Fracao(A + B); Put(" (Esperado: 5 / 6)"); New_Line;
  Put("A - B = "); Imprimir_Fracao(A - B); Put(" (Esperado: 1 / 6)"); New_Line;
  Put("A * B = "); Imprimir_Fracao(A * B); Put(" (Esperado: 1 / 6)"); New_Line;
  Put("A / B = "); Imprimir_Fracao(A / B); Put(" (Esperado: 3 / 2)"); New_Line;
end;

  -- Teste 4: Impressão (b e c)
  New_Line;
  Put_Line("Teste 4: Impressao (b e c)");
declare
  R_Print : Racional := Criar(2, 3);
begin
  Put("Fracao (b): ");
  Imprimir_Fracao(R_Print); -- Saída: 2 / 3
  New_Line;

  Put("Float (c): ");
  Imprimir_Float(R_Print); -- Saída: 0.667
  New_Line;
end;

exception
  when E: Constraint_Error =>
    Put_Line("ERRO FATAL: " & Exception_Information(E));
end Solucao_14;

```

## IMAGEM DA EXECUÇÃO

The screenshot shows the OneCompiler IDE interface. The code editor contains Ada code for a package named Racional\_TD4. The code includes various procedures and functions for creating, reducing, and performing arithmetic operations on rational numbers. The execution results window shows test cases for creation, reduction, addition, subtraction, multiplication, division, and printing of rational numbers.

```
1 -- HelloWorld.adb para Questão 14: TDA_Racional
2 with Ada.Text_IO;           use Ada.Text_IO;
3 with Ada.Integer_Text_IO;   use Ada.Integer_Text_IO;
4 with Ada.Float_Text_IO;    use Ada.Float_Text_IO;
5 with Ada.Exceptions;      use Ada.Exceptions;
6
7 procedure Solucao_14 is
8
9   -- ESPECIFICAÇÃO DO TDA RACIONAL
10
11  package Racional_TD4 is
12
13    type Racional is private;
14
15    -- Função "Construtor de operadores default (0, 1) se nenhum for dado.
16    -- 1. Gerar o formato reduzido e o sinal correto.
17    -- 2. Gerar o formato ponto flutuante.
18    function Crlar(N : Integer := 0; D : Integer := 1) return Racional;
19
20    -- Funções de "get" (óveis para as operações)
21    function Get_Num(R : Racional) return Integer;
22    function Get_Den(R : Racional) return Positive;
23
24    -- a) Funções de aritmética (obrigatória de operadores)
25    function "+"(L, R : Racional) return Racional;
26    function "-"(L, R : Racional) return Racional;
27    function "/"(L, R : Racional) return Racional;
28    function "*"(L, R : Racional) return Racional;
29
30    -- b) Imprimir em formato fração (a / b)
31    procedure Imprimir_Fracao(R : in Racional);
32
33    -- c) Imprimir em formato ponto flutuante
34    procedure Imprimir_Float(R : in Racional);
35
36
37  private
38    -- Parte privada da especificação
39    type Racional is record
40      Numerador : Positive' Positive; -- Para garantir que nunca seja 0 ou negativo.
41      Denominador : Positive := 1;
42    end record;
43
44
45 end Racional_TD4;
```

STDIN  
Input for the program (Optional)

Output:  
--- Teste TDA Racional ---  
Teste 1: Criacao e Reducao  
R1 (10/20) = 1 / 2  
R2 (-7/-21) = -1 / 3  
Teste 2: Construtor Default  
R\_Default = 0 / 1  
Teste 3: Arithmetica (1/2 e 1/3)  
A = 1 / 2  
B = 1 / 3  
A + B = 5 / 6 (Esperado: 5 / 6)  
A - B = 1 / 6 (Esperado: 1 / 6)  
A \* B = 1 / 6 (Esperado: 1 / 6)  
A / B = 3 / 2 (Esperado: 3 / 2)  
Teste 4: Impressao (b e c)  
Fracao (b): 2 / 3  
Float (c): 0.667