

PRÁCTICA 1

Relaciones 1-5.

SISTEMAS DE TIEMPO REAL

3º Ingeniería Informática

Lucas Barrientos Muñoz

Contenido

[Relación 1. 3](#_Toc162119209)

[Ejercicio 1. 3](#_Toc162119210)

[Ejercicio 2. 4](#_Toc162119211)

[Ejercicio 3. 4](#_Toc162119212)

[Ejercicio 4. 5](#_Toc162119213)

[Ejercicio 5. 6](#_Toc162119214)

[Ejercicio 6. 7](#_Toc162119215)

[Main para ejecutar todos los procedimientos. 9](#_Toc162119216)

[Relación 2. 9](#_Toc162119217)

[Ejercicio 1. 9](#_Toc162119218)

[Parte Opcional. 10](#_Toc162119219)

[Main para ejecutar todos los procedimientos. 11](#_Toc162119220)

[Relación 3. 12](#_Toc162119221)

[Leer y escribir matriz. 12](#_Toc162119222)

[Transponer matriz. 13](#_Toc162119223)

[Main para ejecutar todos los procedimientos. 14](#_Toc162119224)

[Relación 4. 14](#_Toc162119225)

[Ejercicio 1. 14](#_Toc162119226)

[Main para ejecutar todos los procedimientos. 16](#_Toc162119227)

[Relación 5. 16](#_Toc162119228)

[Ejercicio 1. 16](#_Toc162119229)

[Main para ejecutar todos los procedimientos. 18](#_Toc162119230)

# Relación 1.

## Ejercicio 1.

ADS

with Ada.Text\_IO; use Ada.Text\_IO;

package Ejercicio1 is

   subtype Tipo\_A is Integer range -120 .. 120;

   subtype Tipo\_B is Integer range 0 .. 50;

   subtype Tipo\_C is Integer range 0 .. 255;

   A : Tipo\_A := 100;

   B : Tipo\_B := 25;

   C : Tipo\_C := 200;

   procedure Operaciones\_Ejercicio1;

end Ejercicio1;

ADB

package body Ejercicio1 is

   procedure Operaciones\_Ejercicio1 is

      Resultado\_AB : Integer;

      Resultado\_AC : Integer;

      Resultado\_BC : Integer;

   begin

      Resultado\_AB := A + B;

      Resultado\_AC := A + C;

      Resultado\_BC := B + C;

      Put\_Line("Ejercicio 1:");

      Put\_Line("Resultado de A + B: " & Integer'Image(Resultado\_AB));

      Put\_Line("Resultado de A + C: " & Integer'Image(Resultado\_AC));

      Put\_Line("Resultado de B + C: " & Integer'Image(Resultado\_BC));

      New\_Line;

   end Operaciones\_Ejercicio1;

end Ejercicio1;

## Ejercicio 2.

ADS

with Ada.Text\_IO; use Ada.Text\_IO;

package Ejercicio2 is

   type Semaforo is (Rojo, Amarillo, Verde);

   procedure Mostrar\_Semaforo\_Ejercicio2(Color : Semaforo);

end Ejercicio2;

ADB

package body Ejercicio2 is

   procedure Mostrar\_Semaforo\_Ejercicio2(Color : Semaforo) is

   begin

      Put\_Line("Ejercicio 2: ");

      case Color is

         when Rojo =>

            Put\_Line("El semaforo esta en rojo.");

         when Amarillo =>

            Put\_Line("El semaforo esta en amarillo.");

         when Verde =>

            Put\_Line("El semaforo esta en verde.");

      end case;

      New\_Line;

   end Mostrar\_Semaforo\_Ejercicio2;

end Ejercicio2;

## Ejercicio 3.

ADS

with Ada.Text\_IO; use Ada.Text\_IO;

package Ejercicio3 is

   type A is delta 0.01 range -50.0 .. 50.0;

   type B is delta 0.001 range -200.0 .. 200.0;

   A\_Variable : A := 25.0;

   B\_Variable : B := 150.123;

   procedure Operaciones\_Ejercicio3;

end Ejercicio3;

ADB

package body Ejercicio3 is

   procedure Operaciones\_Ejercicio3 is

      Resultado : Float;

   begin

      Put\_Line("Ejercicio 3: ");

      Resultado := Float(A\_Variable) + Float(B\_Variable);

      Put\_Line("Resultado de A + B: " & Float'Image(Resultado));

      New\_Line;

   end Operaciones\_Ejercicio3;

end Ejercicio3;

## Ejercicio 4.

ADS

with Ada.Text\_IO; use Ada.Text\_IO;

package Ejercicio4 is

   type A\_Array is array (1 .. 150) of Float;

   type B\_Array is array (1 .. 200, 1 .. 200, 1 .. 200) of Integer;

   type C\_Array is array (Positive range <>) of Float;

   A\_Variable : A\_Array := (others => 0.0);

   B\_Variable : B\_Array := (others => (others => (others => 0)));

   C\_Variable : C\_Array := (0.0, 0.0, 0.0, 0.0, 0.0); -- Ejemplo de inicialización con tamaño 5

   procedure Operaciones\_Ejercicio4;

end Ejercicio4;

ADB

package body Ejercicio4 is

   procedure Operaciones\_Ejercicio4 is

   begin

      Put\_Line("Ejercicio 4: ");

      Put\_Line("Variable A:");

      for I in A\_Variable'Range loop

         Put(A\_Variable(I)'Image & " ");

      end loop;

      New\_Line;

      Put\_Line("Variable B:");

      for I in B\_Variable'Range(1) loop

         for J in B\_Variable'Range(2) loop

            for K in B\_Variable'Range(3) loop

               Put(B\_Variable(I, J, K)'Image & " ");

            end loop;

            New\_Line;

         end loop;

         New\_Line;

      end loop;

      Put\_Line("Variable C:");

      for I in C\_Variable'Range loop

         Put(C\_Variable(I)'Image & " ");

      end loop;

      New\_Line;

   end Operaciones\_Ejercicio4;

end Ejercicio4;

## Ejercicio 5.

ADS

with Ada.Text\_IO; use Ada.Text\_IO;

package Ejercicio5 is

   Constante\_Cadena : constant String := "TIEMPO REAL";

   procedure Operaciones\_Ejercicio5;

end Ejercicio5;

ADB

package body Ejercicio5 is

   procedure Operaciones\_Ejercicio5 is

   begin

      Put\_Line("Ejercicio 5:");

      Put\_Line("La constante de cadena es: " & Constante\_Cadena);

      New\_Line;

   end Operaciones\_Ejercicio5;

end Ejercicio5;

## Ejercicio 6.

ADS

with Ada.Text\_IO; use Ada.Text\_IO;

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

package Ejercicio6 is

   type Fecha\_Nacimiento is record

      Dia   : Integer;

      Mes   : Integer;

      Anio  : Integer;

   end record;

   type Datos\_Personales is record

      Nombre          : Unbounded\_String;

      Apellidos       : Unbounded\_String;

      Fecha\_Nac       : Fecha\_Nacimiento;

   end record;

   Lucas : constant Datos\_Personales := (

      Nombre      => To\_Unbounded\_String("Lucas"),

      Apellidos   => To\_Unbounded\_String("Barrientos Muñoz"),

      Fecha\_Nac   => (Dia => 27, Mes => 7, Anio => 2003)

                                        );

   Adrian : constant Datos\_Personales := (

      Nombre      => To\_Unbounded\_String("Adrian"),

      Apellidos   => To\_Unbounded\_String("Antequera Ramirez"),

      Fecha\_Nac   => (Dia => 12, Mes => 2, Anio => 2003)

                                        );

   procedure Imprimir\_Ejercicio6;

end Ejercicio6;

ADB

with Ada.Text\_IO; use Ada.Text\_IO;

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

package body Ejercicio6 is

   procedure Imprimir\_Ejercicio6 is

   begin

      Put\_Line("Ejercicio 6: ");

      Put\_Line("Nombre: " & To\_String(Lucas.Nombre));

      Put\_Line("Apellidos: " & To\_String(Lucas.Apellidos));

      Put\_Line("Fecha de Nacimiento: " & Lucas.Fecha\_Nac.Dia'Image & "/" &

                                            Lucas.Fecha\_Nac.Mes'Image & "/" &

                                            Lucas.Fecha\_Nac.Anio'Image);

      New\_Line;

      Put\_Line("Nombre: " & To\_String(Adrian.Nombre));

      Put\_Line("Apellidos: " & To\_String(Adrian.Apellidos));

      Put\_Line("Fecha de Nacimiento: " & Adrian.Fecha\_Nac.Dia'Image & "/" &

                                            Adrian.Fecha\_Nac.Mes'Image & "/" &

                                            Adrian.Fecha\_Nac.Anio'Image);

   end Imprimir\_Ejercicio6;

end Ejercicio6;

## Main para ejecutar todos los procedimientos.

with Ejercicio1; use Ejercicio1;

with Ejercicio2; use Ejercicio2;

with Ejercicio3; use Ejercicio3;

with Ejercicio4; use Ejercicio4;

with Ejercicio5; use Ejercicio5;

with Ejercicio6; use Ejercicio6;

procedure Main is

begin

   Operaciones\_Ejercicio1;

   Mostrar\_Semaforo\_Ejercicio2(Verde); -- Introducir por par�metro el color del sem�foro

   Operaciones\_Ejercicio3;

   Operaciones\_Ejercicio4;

   Operaciones\_Ejercicio5;

   Imprimir\_Ejercicio6;

end Main;

# Relación 2.

## Ejercicio 1.

ADS

with Ada.Text\_IO; use Ada.Text\_IO;

package Ejercicio1 is

   procedure Procesar\_Cadena;

end Ejercicio1;

ADB

with Ada.Text\_IO; use Ada.Text\_IO;

with Ejercicio1;

package body Ejercicio1 is

   procedure Procesar\_Cadena is

      Cadena : constant String := "ABCDEFG";

   begin

      for I in Cadena'Range loop

         case Cadena(I) is

            when 'A' | 'B' =>

               Put\_Line("Opción 1");

            when 'C' | 'D' | 'E' =>

               Put\_Line("Opción 2");

            when 'F' =>

               Put\_Line("Opción 3");

            when others =>

               Put\_Line("Otra opción");

         end case;

      end loop;

      New\_Line;

   end Procesar\_Cadena;

end Ejercicio1;

## Parte Opcional.

ADS

with Ada.Text\_IO; use Ada.Text\_IO;

package Ejercicio1\_Opcional is

   procedure Procesar\_Cadena\_Entrada;

end Ejercicio1\_Opcional;

ADB

with Ada.Text\_IO; use Ada.Text\_IO;

with Ejercicio1\_Opcional;

package body Ejercicio1\_Opcional is

procedure Procesar\_Cadena\_Entrada is

   Cadena : String(1 .. 100); -- Permitimos cadenas de hasta 100 caracteres

   Last   : Natural;

   begin

      Put\_Line("Parte Opcional: ");

      New\_Line;

   Put\_Line("Ingrese una cadena:");

   Get\_Line(Cadena, Last);

      for I in Cadena'Range loop

         case Cadena(I) is

            when 'A' | 'B' =>

               Put\_Line("Opción 1");

            when 'C' | 'D' | 'E' =>

               Put\_Line("Opción 2");

            when 'F' =>

               Put\_Line("Opción 3");

            when others =>

               Put\_Line("Otra opción");

         end case;

      end loop;

   end Procesar\_Cadena\_Entrada;

end Ejercicio1\_Opcional;

## Main para ejecutar todos los procedimientos.

with Ejercicio1; use Ejercicio1;

with Ejercicio1\_Opcional; use Ejercicio1\_Opcional;

procedure Main is

begin

   Procesar\_Cadena;

   Procesar\_Cadena\_Entrada;

end Main;

# Relación 3.

## Leer y escribir matriz.

ADS

package Ejercicio1 is

   type Matrix is array(Integer range <>, Integer range <>) of Integer;

   procedure Leer\_Matriz(FileName : String; M2 : out Matrix);

   procedure Escribir\_Matriz(FileName : String; M2 : Matrix);

end Ejercicio1;

ADB

with Ada.Integer\_Text\_IO; use Ada.Integer\_Text\_IO;

with Ada.Text\_IO; use Ada.Text\_IO;

package body Ejercicio1 is

   procedure Leer\_Matriz(FileName : String; M2 : out Matrix) is

      F : File\_Type;

      Rows, Cols : Integer;

   begin

      Open(F, In\_File, FileName);

      Get(F, Rows);

      Get(F, Cols);

      New\_Line(F);

      for I in 1 .. Rows loop

         for J in 1 .. Cols loop

            Get(F, M2(I, J));

         end loop;

         New\_Line(F);

      end loop;

      Close(F);

   end Leer\_Matriz;

   procedure Escribir\_Matriz(FileName : String; M2 : Matrix) is

      F : File\_Type;

      Rows, Cols : Integer;

   begin

      Open(F, Out\_File, FileName);

      Rows := M2'Length(1);

      Cols := M2'Length(2);

      Put(F, Rows);

      Put(F, ' ');

      Put(F, Cols);

      New\_Line(F);

      for I in 1 .. Rows loop

         for J in 1 .. Cols loop

            Put(F, M2(I, J), Width => 3);

            Put(F, ' ');

         end loop;

         New\_Line(F);

      end loop;

      Close(F);

   end Escribir\_Matriz;

end Ejercicio1;

## Transponer matriz.

ADS

package Matriz\_Transpuesta is

   type Matrix is array(Integer range <>, Integer range <>) of Integer;

   procedure Transponer(M : in out Matrix);

end Matriz\_Transpuesta;

ADB

package body Matriz\_Transpuesta is

   procedure Transponer(M : in out Matrix) is

      Temp : Integer;

   begin

      for I in M'Range(1) loop

         for J in M'Range(2) loop

            if J > I then

               Temp := M(I, J);

               M(I, J) := M(J, I);

               M(J, I) := Temp;

            end if;

         end loop;

      end loop;

   end Transponer;

end Matriz\_Transpuesta;

## Main para ejecutar todos los procedimientos.

with Ada.Text\_IO; use Ada.Text\_IO;

with Ejercicio1;

with Matriz\_Transpuesta;

procedure Main is

   M2 : Ejercicio1.Matrix(1 .. 5, 1 .. 10);

   M : Matriz\_Transpuesta.Matrix(1 .. 5, 1 .. 10);

   output : File\_Type;

begin

   Create(output, Out\_File, "output.txt");

   Ejercicio1.Leer\_Matriz("input.txt", M2);

   Matriz\_Transpuesta.Transponer(M);

   Ejercicio1.Escribir\_Matriz("output.txt", M2);

   Close(output);

end Main;

# Relación 4.

## Ejercicio 1.

ADS

package Cola is

   type Elemento is range -100 .. 100;

   function Vacia return Boolean;

   procedure Poner(E: Elemento);

   procedure Quitar(E: out Elemento);

end Cola;

ADB

package body Cola is

   type Nodo;

   type Enlace is access Nodo;

   type Nodo is

      record

         Contenido : Elemento;

         Siguiente : Enlace;

      end record;

   Primero, Ultimo : Enlace := null;

   function Vacia return Boolean is

   begin

      return Primero = null;

   end Vacia;

   procedure Poner(E: Elemento) is

      Nuevo : Enlace;

   begin

      Nuevo := new Nodo;

      Nuevo.Contenido := E;

      Nuevo.Siguiente := null;

      if Vacia then

         Primero := Nuevo;

      else

         Ultimo.Siguiente := Nuevo;

      end if;

      Ultimo := Nuevo;

   end Poner;

   procedure Quitar(E: out Elemento) is

      Viejo : Enlace;

   begin

      Viejo := Primero;

      E := Viejo.Contenido;

      Primero := Viejo.Siguiente;

      if Primero = null then

         Ultimo := null;

      end if;

   end Quitar;

end Cola;

## Main para ejecutar todos los procedimientos.

with Ada.Text\_IO, Ada.Integer\_Text\_IO; use Ada.Text\_IO, Ada.Integer\_Text\_IO;

with Cola; use Cola;

procedure Main is

   E : Elemento;

   N : Integer;

begin

   for i in 1 .. 10 loop

      E := Elemento(i);

      Poner(E);

   end loop;

   Put\_Line("Datos de la cola: ");

   while Vacia = false loop

      Quitar(Elemento(N));

      Put(Integer'Image(N));

   end loop;

end Main;

# Relación 5.

## Ejercicio 1.

ADS

package Numeros\_Complejos is

   type Complejo is private;

   function "+"(X, Y : Complejo) return Complejo;

   function "-"(X, Y : Complejo) return Complejo;

   function "\*"(X, Y : Complejo) return Complejo;

   function "/"(X, Y : Complejo) return Complejo;

   function Conj(X : Complejo) return Complejo;

   function P\_Real(X : Complejo) return Complejo;

   function P\_Imag(X : Complejo) return Complejo;

   function Comp(R, I : Float) return Complejo;

   function numero\_float(X : Complejo) return String;

private type Complejo is

      record

         P\_Real : Float;

         P\_Imag : Float;

      end record;

end Numeros\_Complejos;

ADB

package body Numeros\_Complejos is

   function "+" (X, Y : Complejo) return Complejo is

   begin

      return Complejo'(X.P\_Real + Y.P\_Real, X.P\_Imag + Y.P\_Imag);

   end "+";

   function "-" (X, Y : Complejo) return Complejo is

   begin

      return Complejo'(X.P\_Real - Y.P\_Real, X.P\_Imag - Y.P\_Imag);

   end "-";

   function "\*" (X, Y : Complejo) return Complejo is

      P\_Real : Float := X.P\_Real \* Y.P\_Real - X.P\_Imag \* Y.P\_Imag;

      P\_Imag : Float := X.P\_Real \* Y.P\_Real + X.P\_Imag \* Y.P\_Imag;

   begin

       return Complejo'(P\_Real => P\_Real, P\_Imag => P\_Imag);

   end "\*";

   function "/" (X, Y : Complejo) return Complejo is

      P\_Real : Float := (X.P\_Real \* X.P\_Real + X.P\_Imag \* Y.P\_Imag) / (Y.P\_Real \*\* 2 + Y.P\_Imag \*\* 2);

      P\_Imag : Float := (X.P\_Imag \* Y.P\_Real + X.P\_Real \* Y.P\_Imag) / (Y.P\_Real \*\* 2 + Y.P\_Imag \*\* 2);

   begin

      return Complejo'(P\_Real => P\_Real, P\_Imag => P\_Imag);

   end "/";

   function Conj (X : Complejo) return Complejo is

   begin

      return Complejo'(P\_Real => X.P\_Real, P\_Imag => -X.P\_Imag);

   end Conj;

   function P\_Real (X : Complejo) return Complejo is

   begin

      return Comp(X.P\_Real, 0.0);

   end P\_Real;

   function P\_Imag (X : Complejo) return Complejo is

   begin

      return Comp(X.P\_Imag, 0.0);

   end P\_Imag;

   function Comp (R, I : Float) return Complejo is

      C : Complejo;

   begin

      C.P\_Real := R;

      C.P\_Imag := I;

      return C;

   end Comp;

   function numero\_float (X : Complejo) return String is

   begin

      return Float'Image(X.P\_Real) & " + " & Float'Image(X.P\_Imag) & "i";

   end numero\_float;

end Numeros\_Complejos;

## Main para ejecutar todos los procedimientos.

with Ada.Text\_IO, Numeros\_Complejos; use Ada.Text\_IO, Numeros\_Complejos;

procedure Main is

   x, y, Suma, Resta, Multiplicacion, Division, Conjugado : Complejo;

begin

   x := Comp(5.0, 3.0);

   y := Comp(4.0, 9.0);

   Suma := x + y;

   Resta := x - y;

   Multiplicacion := x \* y;

   Division := x / y;

   Conjugado := Conj(x);

   Put\_Line("Suma = " & numero\_float(Suma));

   Put\_Line("Resta = " & numero\_float(Resta));

   Put\_Line("Multiplicación = " & numero\_float(Multiplicacion));

   Put\_Line("División = " & numero\_float(Division));

   Put\_Line("Conjugado = " & numero\_float(Conjugado));

end Main;