**Ejercicios Bloque 1**

Sistemas de Tiempo Real

**Grupo 17**

Pablo Gómez Rivas

Jesús Fornieles Muñoz

Universidad de Almería

Almería, domingo 24 de marzo de 2024

**Índice de Contenidos** Página

Ejercicio 1: Tipos de Datos en ADA ……….3

Ejercicio 1.1: Tipos de Datos Enteros 3

Ejercicio 1.2: Tipos de Datos Discretos 4

Ejercicio 1.3: Tipos de Datos Reales 4

Ejercicio 1.4: Arrays 6

Ejercicio 1.5: Cadenas 7

Ejercicio 1.6: Registros 7

Ejercicio 2: Estructuras de Control en ADA ……….8

Ejercicio 2.1: Instrucciones y Estructuras de Control 8

Ejercicio 3: Uso de Ficheros en ADA ……….9

Ejercicio 3.1: Fichero Transpuesto 9

Ejercicio 4: Ocultación de Información en ADA ……...10

Ejercicio 4.1: Estructuras Básicas de Datos 10

Ejercicio 5: Sobrecarga de Operadores en ADA ……...13

Ejercicio 5.1: Paquete de Números Complejos 13

# Ejercicio 1: Tipos de Datos en ADA

## Interfaz de usuario gráficaEjercicio 1.1: Tipos de Datos Enteros

* **Importar las bibliotecas necesarias:** En la primera línea del código, se importa el paquete Ada.Text\_IO, que proporciona procedimientos y funciones para entrada/salida de texto. La cláusula use Ada.Text\_IO permite usar los procedimientos y funciones de este paquete sin calificarlos con el prefijo Ada.Text\_IO.
* **Definición de subtipos:** Se definen tres subtipos de enteros, tipoA, tipoB y tipoC, con rangos específicos. Estos subtipos limitan los valores que pueden tomar las variables declaradas con estos tipos.
* **Declaración de variables:** Se declaran las variables A, B y C, cada una con su respectivo subtipo (tipoA, tipoB y tipoC, respectivamente), y se declaran tres variables adicionales sol1, sol2 y sol3 de tipo Integer para almacenar los resultados de las operaciones.
* **Asignación de valores a las variables:** Se asignan valores específicos a las variables A, B y C. En este caso, A se asigna como 8, B como 16 y C como 32.
* **Realización de operaciones aritméticas:** Se realizan tres operaciones aritméticas simples: la suma de A y B, la suma de A y C, y la suma de B y C. Los resultados de estas operaciones se almacenan en las variables sol1, sol2 y sol3, respectivamente.
* **Impresión de resultados:** Se utiliza el procedimiento Put\_Line para imprimir los resultados de las operaciones en la salida estándar. Cada línea imprime una cadena que representa la operación realizada y el resultado correspondiente, utilizando la función Integer'Image para convertir el resultado entero en una cadena legible para la salida.

## Texto Descripción generada automáticamenteEjercicio 1.2: Tipos de Datos Discretos

* **Declaración del tipo enumerado:** Se define el tipo enumerado Semaforo que tiene tres posibles valores: Rojo, Amarillo, y Verde.
* **Declaración de la variable:** Se declara una variable Sem del tipo Semaforo.
* **Asignación de valores:** Se asignan valores a la variable Sem utilizando el atributo 'Val' del tipo enumerado Semaforo. Se asignan los valores 0, 1 y 2 respectivamente, que corresponden a los valores Rojo, Amarillo, y Verde según la declaración del tipo enumerado.
* **Impresión de valores:** Se utilizan las funciones Put\_Line y 'Image' para imprimir en la consola los valores asignados a Sem. 'Image' convierte el valor de Sem en su representación de cadena correspondiente.

## TextoEjercicio 1.3: Tipos de Datos Reales

* **Define dos tipos de datos:** TipoA y TipoB. TipoA es un tipo decimal con precisión de 2 decimales y rango de -50.00 a 50.00. TipoB es un tipo decimal con precisión de 3 dígitos y rango de -200.0 a 200.0.
* **Declara tres variables:** A de tipo TipoA, B de tipo TipoB y Sol1 de tipo Float.
* **Asigna valores iniciales a las variables A y B:** A se inicializa con 1.00 y B con 2.00.
* **Realiza la suma de A y B** después de convertirlos a tipo Float para evitar pérdida de precisión. Esto se hace mediante la conversión explícita de A y B a Float y luego se realiza la suma.
* **Imprime el resultado de la suma** en la consola utilizando la función 'Put\_Line' para mostrar el resultado como una cadena de caracteres, obteniendo la representación en texto de Sol1 mediante la función 'Float'Image'.

## Texto Descripción generada automáticamenteEjercicio 1.4: Arrays

* **Define varios tipos de datos:** Vector1 como un arreglo de 150 elementos de tipo Float, Matriz1 como un arreglo tridimensional de tipo Float y Vector2 como un arreglo de enteros de longitud variable de tipo Float.
* **Declara variables de los tipos definidos anteriormente:** V1 como un Vector1, M1 como una Matriz1 y V2 como un Vector2 con un rango de 0 a 10.
* **Inicializa las variables V1, M1 y V2** con valores predeterminados de 0.0.
* **Utiliza bucles para imprimir** los elementos de V1, M1 y V2 en la consola.
* **Imprime los elementos de V1** mediante un bucle for que recorre todos los índices del arreglo V1.
* **Imprime los elementos de M1** mediante bucles anidados for que recorren todos los índices de la matriz M1.
* **Imprime los elementos de V2** mediante un bucle for que recorre todos los índices del arreglo V2.

## TextoEjercicio 1.5: Cadenas

* **Declaración del paquete:** Se incluye el paquete Ada.Text\_IO, que proporciona procedimientos y funciones para entrada/salida de texto en Ada.
* **Uso del paquete:** Se usa la cláusula use para utilizar directamente los procedimientos y funciones del paquete Ada.Text\_IO sin tener que calificarlos con el prefijo del paquete.
* **Declaración de la variable:** Se declara una variable c de tipo String que contiene el valor "TIEMPO REAL".
* **Impresión en la consola:** Dentro del bloque begin y end, se utiliza el procedimiento Put\_Line del paquete Ada.Text\_IO para imprimir la cadena de texto contenida en la variable c en una nueva línea en la consola.

## TextoEjercicio 1.6: Registros

* **Declaración de subtipos y tipos de registro:** Se definen subtipos para representar días, meses y años, y se define un tipo de registro llamado Fecha que contiene los campos dia, Mes y Ano. Además, se define otro tipo de registro llamado Datos\_Personales que contiene campos para el nombre, apellidos y una fecha de tipo Fecha.
* **Declaración de una variable de tipo Datos\_Personales:** Se declara una variable llamada P1 de tipo Datos\_Personales.
* **Inicialización de la variable P1:** Se inicializa la variable P1 con los valores proporcionados para el nombre, apellidos y fecha de nacimiento.
* **Acceso a los campos del registro:** Se accede a los campos del registro P1 y se imprime el nombre, apellidos y fecha de nacimiento en la consola.

# Ejercicio 2: Estructuras de Control en ADA

## Texto Descripción generada automáticamenteEjercicio 2.1: Instrucciones y Estructuras de Control

* **Se importan las bibliotecas necesarias:** Ada.Text\_IO para entrada/salida de texto, Ada.Strings.Unbounded para trabajar con cadenas de longitud variable y Ada.Text\_IO.Unbounded\_IO para entrada/salida de cadenas no limitadas.
* Se declara una variable Input\_String de tipo Unbounded\_String para almacenar la cadena ingresada por el usuario.
* **Se muestra un mensaje** pidiendo al usuario que ingrese una cadena utilizando Put.
* **Se lee la cadena ingresada** por el usuario utilizando Get\_Line y se almacena en Input\_String.
* **Se inicia un bucle for** que recorre cada carácter de la cadena ingresada utilizando Length(Input\_String) como límite.
* Dentro del bucle, se utiliza una declaración case para evaluar cada carácter de la cadena:
  + Si el carácter es 'A' o 'B', se imprime "Opción 1".
  + Si el carácter es 'C', 'D' o 'E', se imprime "Opción 2".
  + Si el carácter es 'F', se imprime "Opción 3".
  + Si el carácter no coincide con ninguno de los anteriores, se imprime "Otra opción".

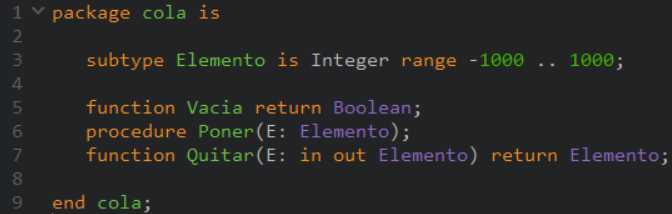
# Ejercicio 3: Uso de Ficheros en ADA

## Ejercicio 3.1: Fichero Transpuesto

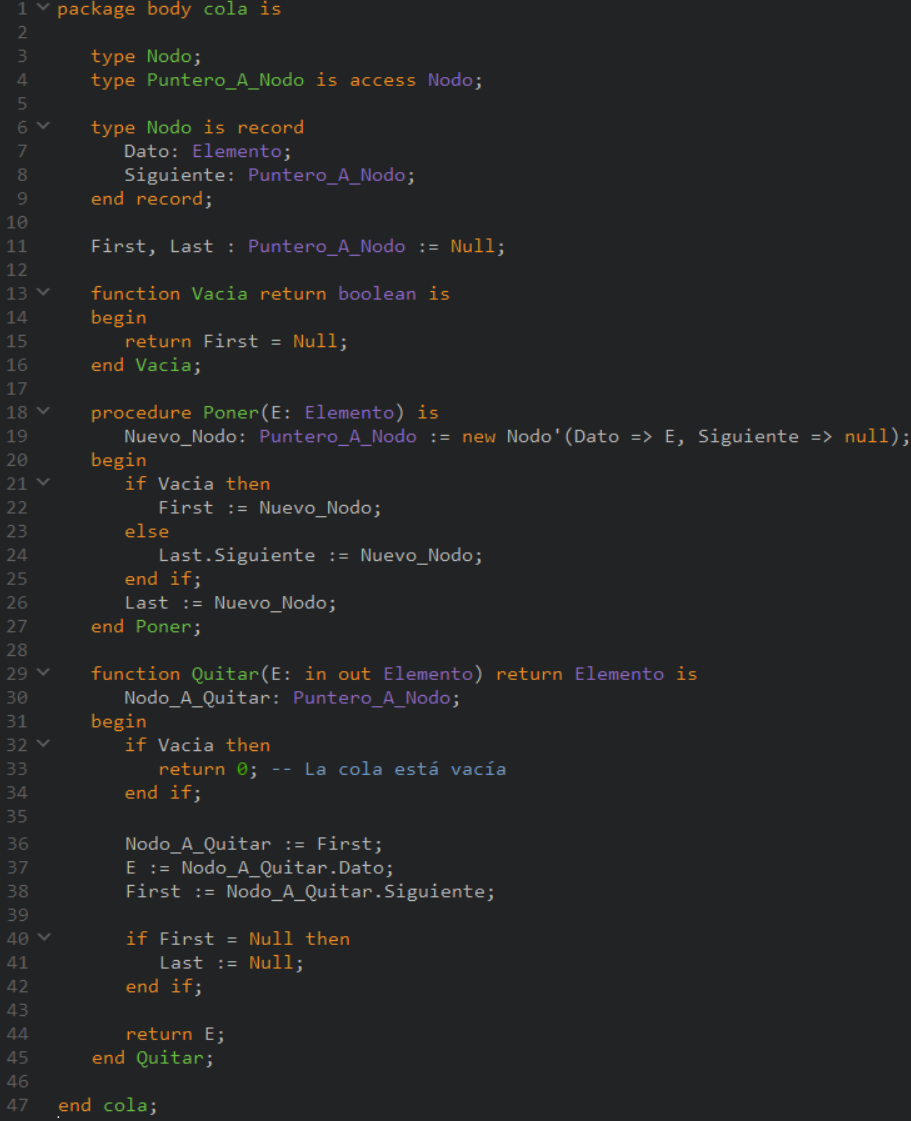
* Abre un archivo de entrada llamado "input.txt".
* Lee los valores del archivo de entrada y los almacena en una matriz bidimensional de tamaño 5x10 llamada "m".
* Cierra el archivo de entrada.
* Calcula la matriz transpuesta de "m" y la almacena en una matriz bidimensional de tamaño 10x5 llamada "mt".
* Abre un archivo de salida llamado "output.txt".
* Escribe la matriz transpuesta "mt" en el archivo de salida.
* Cierra el archivo de salida.

# Ejercicio 4: Ocultación de Información en ADA

## Ejercicio 4.1: Estructuras Básicas de Datos

cola.ads

Se definen el tipo Elemento como entero en el rango de -1000 a 1000, y las funciones Vacia, Poner y Quitar.

cola.adb

* **Se define un tipo de registro llamado "Nodo"** que representa un nodo en la estructura de datos de la cola. Cada nodo contiene un campo "Dato" de tipo "Elemento" y un campo "Siguiente" que es un puntero al siguiente nodo en la cola.
* **Se define un tipo de acceso "Puntero\_A\_Nodo"** que es un puntero a un nodo.
* **Se declaran dos punteros "First" y "Last"** que apuntan al primer y último nodo de la cola respectivamente. Inicialmente, ambos punteros se establecen en "Null" ya que la cola está vacía.
* **Se define una función "Vacia"** que devuelve "True" si la cola está vacía (es decir, si "First" es "Null").
* **Se define un procedimiento "Poner"** que agrega un nuevo elemento a la cola. Este procedimiento crea un nuevo nodo con el elemento dado y lo agrega al final de la cola actualizando los punteros "Last" y "First" según corresponda.
* **Se define una función "Quitar"** que elimina y devuelve el primer elemento de la cola. Esta función primero verifica si la cola está vacía. Si no lo está, guarda el elemento del primer nodo en una variable temporal, actualiza el puntero "First" para apuntar al siguiente nodo y libera el nodo eliminado de la memoria. Si después de la eliminación la cola queda vacía, también actualiza el puntero "Last".

Texto

Descripción generada automáticamente

main4\_1.adb

* Se declara una variable auxiliar llamada "Dato" de tipo "Elemento". Este tipo de dato está definido en el módulo "cola" en el rango de -1000 a 1000.
* Se verifica si la cola está vacía utilizando la función "Vacia" del módulo "cola". Si está vacía, se imprime un mensaje indicando que la cola está vacía.
* Se insertan datos en la cola utilizando la función "Poner" del módulo "cola".
* Se elimina y muestra cada dato de la cola uno por uno en un bucle. Se utiliza la función "Quitar" del módulo "cola" para quitar un dato de la cola. El bucle continúa hasta que la función "Quitar" devuelva 0, lo que indica que la cola está vacía. Se utiliza la función "Put\_Line" para imprimir el dato obtenido de la cola. La función "Integer'Image" se utiliza para convertir el dato a su representación en cadena antes de imprimirlo.

# Ejercicio 5: Sobrecarga de Operadores en ADA

## Texto Descripción generada automáticamenteEjercicio 5.1: Paquete de Números Complejos

numeros\_complejos.ads

Texto

Descripción generada automáticamenteSe define el tipo Complejo que contiene un dato Real y otro Imaginario, ambos float. Se declaran las funciones utilizadas.

numeros\_complejos.adb

* **Declaración del paquete:** Comienza declarando el paquete Numeros\_Complejos.
* **Implementación de las funciones:**
  + **Suma:** Define una función para sumar dos números complejos.
  + **Resta:** Implementa una función para restar dos números complejos.
  + **Producto:** Define una función para multiplicar dos números complejos.
  + **División:** Implementa una función para dividir dos números complejos.
  + **Conjugado:** Define una función para obtener el conjugado de un número complejo.
* **Implementación de cada función:** Cada función toma dos números complejos como parámetros y realiza la operación correspondiente.
* **Retorno de resultados:** Cada función retorna un nuevo número complejo con el resultado de la operación aplicada.

Texto

Descripción generada automáticamentemain5\_1.adb

* **Importación de paquetes:** Se importan los paquetes necesarios, como Ada.Text\_IO para operaciones de entrada/salida y el paquete Numeros\_Complejos que contiene las operaciones con números complejos.
* **Declaración de variables:** Se declaran las variables A, B y Resultado de tipo Complejo, que representan números complejos.
* **Definición de números complejos:** Se definen dos números complejos, A y B, con valores específicos para la parte real e imaginaria.
* **Operaciones con números complejos:** Se realizan varias operaciones con los números complejos definidos:

a. Suma de A y B.

b. Resta de A y B.

c. Producto de A y B.

d. División de A y B.

e. Cálculo del conjugado de A.

* **Impresión de resultados:** Se imprimen los resultados de cada operación realizada, mostrando la parte real e imaginaria de los números complejos obtenidos.