# PRÁCTICA 0
# SISTEMAS EN TIEMPO REAL

Carlos Eguren Esteban

GRADO EN INGENIERÍA DE COMPUTADORES carlos.eguren@edu.uah.es

# Tabla de contenido

# Ejercicios iniciales

Una vez creado el archivo “numeros.adb”, abrimos un terminal en el mismo directorio. A continuación, escribimos la orden “gnatmake numeros.adb”. Con esto conseguimos compilar, encuadernar y enlazar este archivo para después ejecutarlo gracias al comando “./numeros”.

# Ejercicio 4

El procedimiento “Put” que se llama en la línea 26 (en nuestro archivo adjunto a la entrega es la línea 37) pertenece al paquete Ada.Text_IO, que se incluyó al inicio del programa en la línea 5:

```
with Ada.Text_IO; use Ada.Text_IO;
```

# Ejercicio 5

Para poder utilizar “Put(I+J)” sin problemas, añadimos una cláusula “use” para el paquete “Ent_Es” después de su declaración. Esto nos permite llamar directamente a “Put” sin especificar “Ent_Es.Put”.

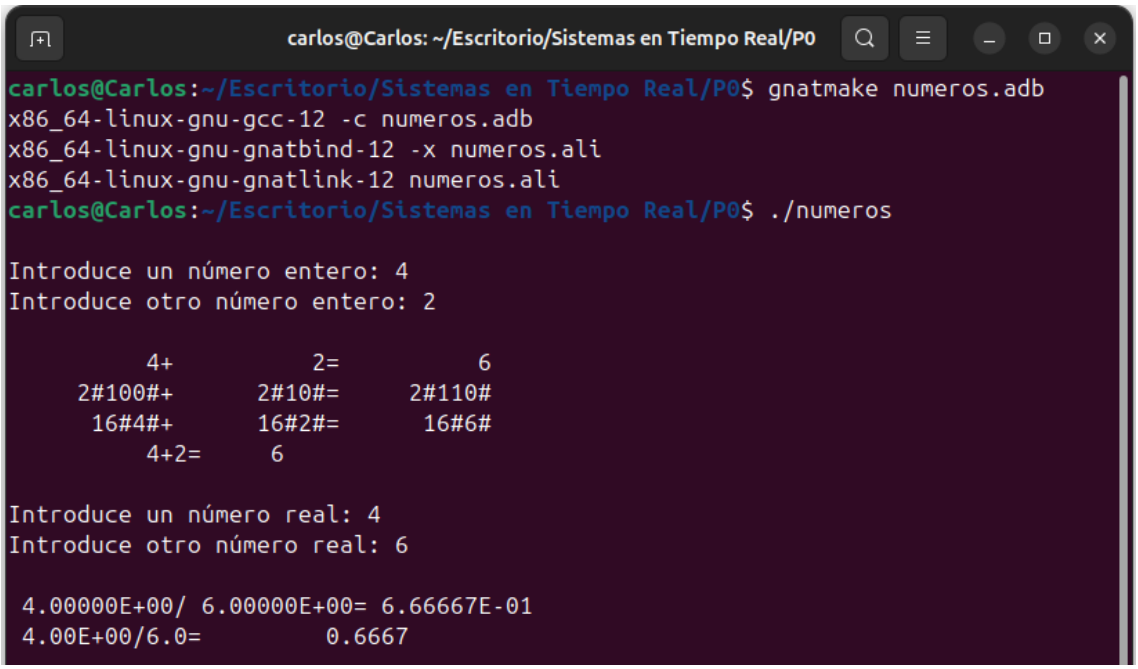
```
13    package Ent_Es is new Ada.Text_Io.Integer_Io(Integer);
14    package Real_Es is new Ada.Text_Io.Float_Io(Float);
15
16    -- Añadimos la cláusula use para Ent_Es
17    use Ent_Es;
18
19    -- Declaración de variables locales.
20    I, J: Integer;
```

Escribimos en la línea 34 (en nuestro código ahora es la línea 36) lo siguiente:

```
Put (I+J);
```

En nuestro archivo final se ha modificado únicamente la línea descrita anteriormente, aunque se pueda realizar exactamente la misma operación en otras líneas del código.

Y volvemos a ejecutar el programa para comprobar que funciona correctamente:




# Ejercicio 6

Yo creo que Ada sabe a qué procedimiento “Put” llamar gracias al tipo de dato que se le pasa por parámetro, siempre y cuando se hayan especificado varias sentencias “use” y tenga varias opciones en las que elegir.

# Ejercicios 7 y 8

Creamos las variables especificadas.

```
-- Ejercicio 7: Declaración de un tipo discreto enumerado
type Luz_de_Trafico_t is (Rojo, Ambar, Verde);

-- Ejercicio 8: Declaración de variable e inicialización
semaforo: Luz_de_Trafico_t := Rojo; -- Variable declarada e inicializada
```

# Ejercicio 9

Creamos el paquete especificado:

```
-- Ejercicio 9: Creacion de paquete
package Sem_Es is new Ada.Text_Io.Enumeration_Io(Luz_de_Trafico_t);
```

# Ejercicio 10

Mostramos por pantalla el valor de la variable con las siguientes instrucciones en el código del programa:

```
-- Ejercicio 10: Uso del paquete Sem_ES para mostrar el estado del semáforo
Put("El estado del semáforo es: ");
Sem_ES.Put(semaforo);
New_Line; New_Line;
```

Ejecutamos el programa de nuevo para comprobar su funcionamiento.

```
carlos@Carlos:~/Escritorio/Sistemas en Tiempo Real/P0$ gnatmake numeros.adb
x86_64-linux-gnu-gcc-12 -c numeros.adb
x86_64-linux-gnu-gnatbind-12 -x numeros.ali
x86_64-linux-gnu-gnatlink-12 numeros.ali
carlos@Carlos:~/Escritorio/Sistemas en Tiempo Real/P0$ ./numeros

Introduce un número entero: 2
Introduce otro número entero: 3

          2+           3=           5
      2#10#+       2#11#=      2#101#
      16#2#+       16#3#=       16#5#
          2+3=      5

Introduce un número real: 2.2
Introduce otro número real: 1.1

 2.20000E+00/ 1.10000E+00= 2.00000E+00
 2.20E+00/1.1=          2.0000

El estado del semáforo es: ROJO
```

Como podemos observar, el programa muestra por pantalla el valor de la variable.

El código del programa se podría simplificar si utilizásemos la cláusula “use” con todos los paquetes del código:

```
use Real_Es;

use Sem_Es;
```

Esto nos permitiría no tener que especificar el paquete al hacer:

```
Put(X);  -- siendo X un número real

Put(semaforo);
```