# Prog. Básica - Laboratorio 1 Introducción a la algorítmica

**Nombre: Markel Ferro Fecha: 25/09/2020**

Establece para cada uno de los siguientes enunciados:

a) si es ambiguo el enunciado.

b) la correspondiente especificación.

c) una lista de casos de prueba.

d) el diseño de un algoritmo para llevarlo a cabo.

**1. Diferencia.** Crea un algoritmo que, dados dos números enteros num1 y num2, calcule su diferencia.

**2. Producto.** Crea un algoritmo que pida al usuario dos números enteros, calcule su producto y lo muestre en pantalla.

**3. La edad aparente.** Crea un algoritmo que, dado un número natural que indica la edad de una persona, calcule su edad aparente. La edad aparente de una persona se calcula obteniendo el doble de su edad aumentado en cuatro.

**4. Área del círculo.** Crea un algoritmo que, dada la longitud de un segmento, calcule el área de un círculo con un radio de dicha longitud.

**5. Es\_par.** Crea un algoritmo que, dado un número natural, indique si es par (true) o impar (false).

**6. Ordenar dos números (I).** Crea un algoritmo que, dados dos valores enteros mayores que 0, num1 y num2, los ordene de menor a mayor de manera que num1 contenga el menor y num2 el mayor (utiliza como base el algoritmo de calcular el máximo –Máximo1– que se explicó en la clase de teoría)

**7. Ordenar dos números (reusando el ejercicio 1.6).** Crea un algoritmo que pida al usuario dos valores enteros mayores que 0 los ordene de menor a mayor usando el algoritmo del ejercicio 3 y los presente por pantalla en ese orden (utiliza como base el algoritmo de calcular el máximo –Máximo2– que se explicó en la clase de teoría).

**8. Ordenar tres números (I).** Crea un algoritmo que pida al usuario tres valores enteros mayores que 0, num1, num2 y num3, los ordene de menor a mayor y los presente por pantalla en ese orden.

1. **Diferencia**

Crea un algoritmo que dados dos números enteros num1 y num2, calcule su diferencia.

* 1. Especificación

Antes que nada, ¿es ambiguo el enunciado?

Después, rellena la especificación:

|  |
| --- |
| Datos/Entrada:  Pre:    Resultado/Salida:  Post: |

* 1. Casos de Prueba

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| # | Num1 | Num2 | Descripción | Diferencia |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |

* 1. Algoritmo

1. **Producto**

Crea un algoritmo que pida al usuario dos números enteros, calcule su producto y lo muestre en pantalla.

* 1. Especificación

Antes que nada, ¿es ambiguo el enunciado? No, ya que el orden de los factores no altera al producto, pero es recomendable proporcionar las fórmulas

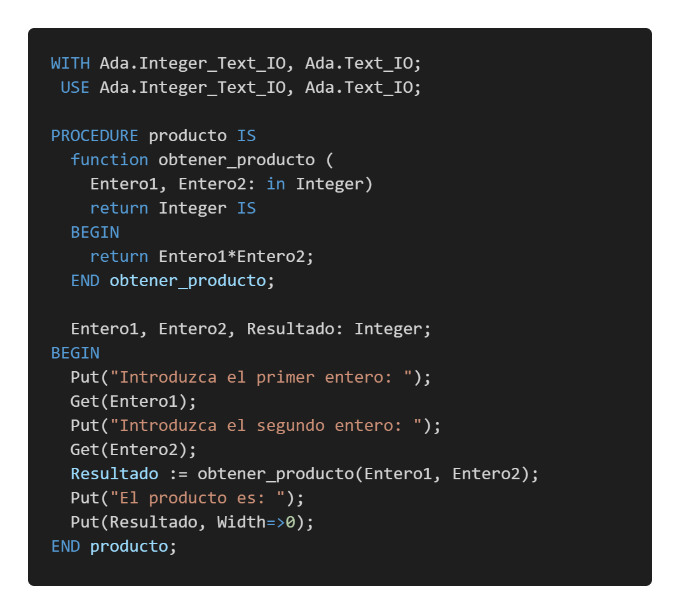
Después, rellena la especificación:

|  |
| --- |
| Datos/Entrada: Entero1, Entero2: Integer  Pre: Entero1 y Entero2 son enteros    Resultado/Salida: Resultado: Integer  Post: Resultado = entero1\*entero2 |

* 1. Casos de Prueba

|  |  |  |  |
| --- | --- | --- | --- |
| # | Entrada estándar (teclado) | Descripción | Salida estándar (pantalla) |
| 0 | 3 1 | La multiplicación de 3\*1 | 3 |
| 1 | -2 5 | La multiplicación con un número negativo | -10 |
| 2 | 5 -2 | La multiplicación con un número negativo | -10 |
| 3 | -6 -6 | La multiplicación de dos números negativos | 36 |
| 4 | 5 0 | La multiplicación con un cero | 0 |
| 5 | 0 5 | La multiplicación con un cero | 0 |
| 6 | 0 0 | La multiplicación dos ceros | 0 |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

* 1. Algoritmo



1. **La edad aparente**

Crea un algoritmo que, dado un número natural que indica la edad de una persona, calcule su edad aparente. La edad aparente de una persona se calcula obteniendo el doble de su edad aumentado en cuatro.

* 1. Especificación

Antes que nada, ¿es ambiguo el enunciado?

Después, rellena la especificación:

|  |
| --- |
| Datos/Entrada:  Pre:    Resultado/Salida:  Post: |

* 1. Casos de Prueba

|  |  |  |  |
| --- | --- | --- | --- |
| # | Edad | Descripción | Edad\_aparente |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

* 1. Algoritmo

1. **Área del círculo**

Crea un algoritmo que, dada la longitud de un segmento, calcule el área de un círculo con un radio de dicha longitud.

* 1. Especificación

Antes que nada, ¿es ambiguo el enunciado?

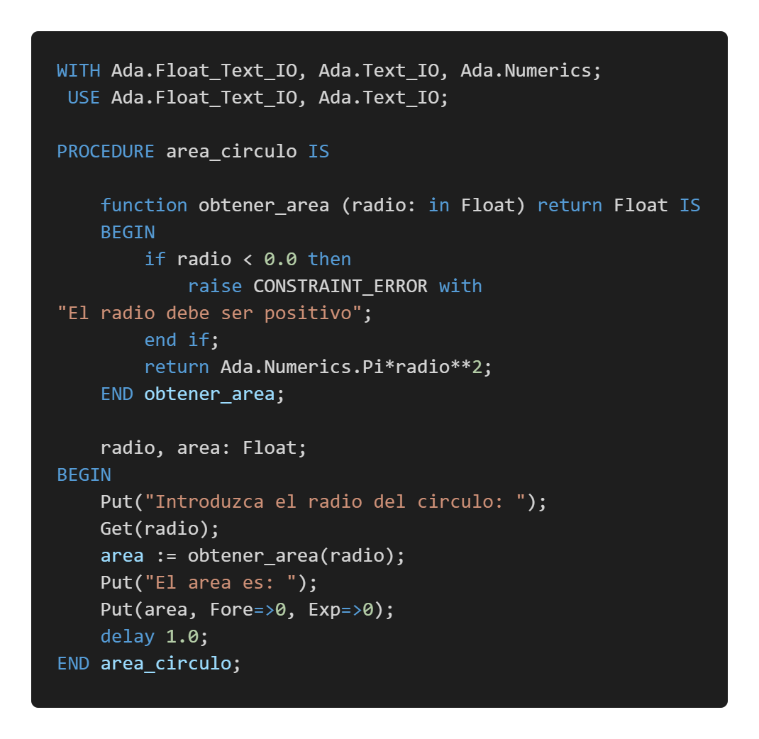
Después, rellena la especificación:

|  |
| --- |
| Datos/Entrada: Radio: Float;  Pre: -- Radio > 0    Resultado/Salida: Area: Float;  Post: -- Area = pi\*r^2 |

* 1. Casos de Prueba

|  |  |  |  |
| --- | --- | --- | --- |
| # | Radio | Descripción | Área |
| 0 | 5 | Debería ser 25pi | 78.5398 |
| 1 | -5 | Debería dar CONSTRAINT ERROR |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

* 1. Algoritmo



1. **Es\_par**

Crea un algoritmo que, dado un número natural, indique si es par (true) o impar (false).

* 1. Especificación

Antes que nada, ¿es ambiguo el enunciado? No

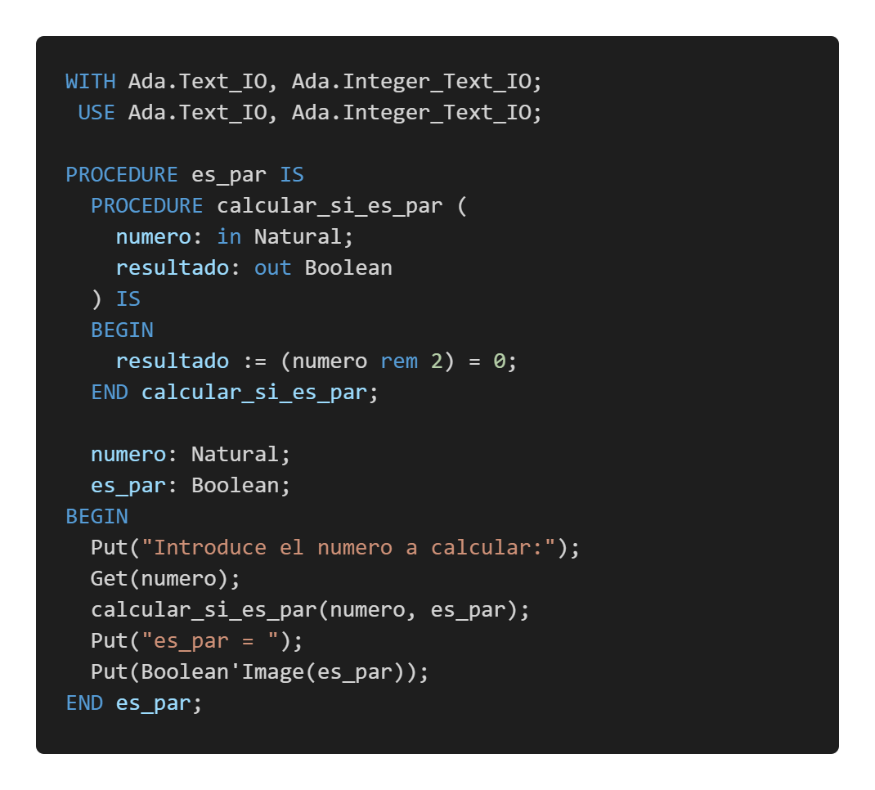
Después, rellena la especificación:

|  |
| --- |
| Datos/Entrada: Numero: Natural  Pre: Es natural    Resultado/Salida: Es\_par: Boolean  Post: -- Si Numero es\_par => Es\_par = true. Si no, es\_par => false |

* 1. Casos de Prueba

|  |  |  |  |
| --- | --- | --- | --- |
| # | Num | Descripción | Es\_par |
| 0 | 1 | 1 es impar | false |
| 1 | 2 | 2 es par | true |
| 2 | 0 | 0 es par (pero siempre es interesante meterlo) | true |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

* 1. Algoritmo



1. **Ordenar dos números (I)**

Crea un algoritmo que, dados dos valores enteros mayores que 0, *num1* y *num2*, los ordene de menor a mayor de manera que *num1* contenga el menor y *num2* el mayor (utiliza como base el algoritmo de calcular el máximo –*Máximo1*– que se explicó en la clase de teoría)

* 1. Especificación

Antes que nada, ¿es ambiguo el enunciado?

Después, rellena la especificación:

|  |
| --- |
| Datos/Entrada: Num1, Num2: Positive  Pre: Los valores de Num1 y Num2 no están ordenados    Resultado/Salida: Num1, Num2: Positive  Post: Los valores de Num1 y Num2 están ordenados de menor a mayor |

* 1. Casos de Prueba

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| # | Num1 | Num2 | Descripción | Num1 | Num2 |
| 0 | 1 | 2 | Los números se quedan en sus respectivas variables | 1 | 2 |
| 1 | 2 | 1 | Los números invierten los valores en sus posiciones | 1 | 2 |
| 2 | 3 | 3 | Los números se quedan si son iguales | 3 | 3 |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |

* 1. Algoritmo

Texto

Descripción generada automáticamente

1. **Ordenar dos números (reusando el ejercicio 6)**

Crea un algoritmo que pida al usuario dos valores enteros mayores que 0 los ordene de menor a mayor usando el algoritmo del ejercicio 3 y los presente por pantalla en ese orden (utiliza como base el algoritmo de calcular el máximo –*Máximo2*– que se explicó en la clase de teoría).

* 1. Especificación

Antes que nada, ¿es ambiguo el enunciado?

Después, rellena la especificación:

|  |
| --- |
| Datos/Entrada:  Pre:    Resultado/Salida:  Post: |

* 1. Casos de Prueba

|  |  |  |  |
| --- | --- | --- | --- |
| # | Entrada estándar (teclado) | Descripción | Salida estándar (pantalla) |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

* 1. Algoritmo

1. **Ordenar tres números (I)**

Crea un algoritmo que pida al usuario tres valores enteros mayores que 0, num1, num2 y num3, los ordene de menor a mayor y los presente por pantalla en ese orden.

* 1. Especificación

Antes que nada, ¿es ambiguo el enunciado? No

Después, rellena la especificación:

|  |
| --- |
| Datos/Entrada: Número1, Número2, Número3: in Positive  Pre: Los tres números ordenados    Resultado/Salida:  Post: |

* 1. Casos de Prueba

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| # | Num1 | Num2 | Num3 | Descripción | Salida estándar (pantalla) |
| 0 | 1 | 2 | 3 | Num1<Num2<Num3 | 3, 2, 1 |
| 1 | 2 | 1 | 3 | Num2<Num1<Num3 | 3, 2, 1 |
| 2 | 3 | 2 | 1 | Num3<Num2<Num1 | 3, 2, 1 |
| 3 | 3 | 1 | 2 | Num2< Num3<Num1 | 3, 2, 1 |
| 4 | 1 | 3 | 2 | Num1<Num3<Num2 | 3, 2, 1 |
| 5 | 2 | 3 | 1 | Num3<Num1<Num2 | 3, 2, 1 |
| 6 |  |  |  |  |  |
| 7 |  |  |  |  |  |
| 8 |  |  |  |  |  |
| 9 |  |  |  |  |  |

* 1. Algoritmo