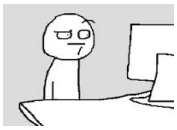


Universidad del País Vasco Euskal Herriko Unibertsitatea

Programación Básica - Laboratorio 4
Secuencias + ADA y Python

Nombre: Iker Fernández y María Fernández **Fecha**: 02/10/2023

1º ejercicio

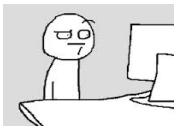
Dada una secuencia de 10 enteros desarrollar un algoritmo tal que imprima por pantalla el último elemento par de esa secuencia y la posición en la que se encuentra.

1. Especificación

Entrada: *una secuencia de 10 números enteros*
Pre: *la secuencia estará totalmente llena*
Salida: *2 valores enteros*
Post: *valor1 | valor $\in$ secuencia inicial ^ valor es el último valor par (empezando por la posición 1)*
valor2| valor2 es la posición de dicho elemento par (será 0 si todos los elementos fuesen impares)

2. Casos de prueba

Entrada	Salida
1,2,4,5,6,8,10,7,7,1	10,7
1,1,1,1,1,1,1,1,1,1	???,0
2,2,2,2,2,2,2,2,2,2	2,10



3. Algoritmo

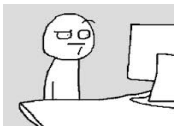
```
secuencia: 10 Integer;
vuelta,num: Integer;
par: Boolean;
escribir("Escribe 10 números");
leer_secuencia(secuencia);
colocarnos_al_final(secuencia);
par←False;
vuelta←10;

repetir salir si par y fuera(secuencia)
        si elemento_actual rem 2=0 entonces
                par←True;
                num←elemento_actual;
        si no
                vuelta←vuelta-1;
        fin_si;
        retroceder(secuencia);
fin_repetir;
escribir (num"," vuelta);
```

4. Simulación

[Para el caso de prueba que queráis, pero indicad cuál.]

Secuencia: (1,2,4,5,6,7,10,7,7,3)

¿Entra?	Elemento_actual (n)	Nº Vueltas	Pos	Par	Salida
Inicializaciones	3	0	10	False	-
SÍ	3	1	10	False	-
SÍ	7	2	9	False	-
SÍ	7	3	8	False	-
SÍ	10	4	7	True	-
NO	10	-	7	True	10,7



2º ejercicio

Dada una secuencia de 10 enteros desarrollar un algoritmo tal que nos diga de entre los números de esa secuencia cuáles son múltiplos del último número de esa secuencia.

1. Especificación

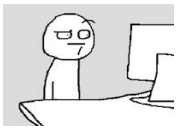
Entrada: *una secuencia de 10 números enteros*
Pre: *la secuencia estará totalmente llena, y el último valor es distinto de 0*
Salida: *uno o más valores enteros*
Post: *valor(es) | valor(es) ∈ secuencia inicial ^ valor(es) resto último valor =0*

2. Casos de prueba

Entrada	Salida
8,5,2,3,6,7,8,1,2,2	8,2,6,8,2,2
1,1,1,1,1,1,1,1,1,3	3
1024,512,256,128,64,32,16,8,4,2	1024,512,256,128,64,32,16,8,4,2

3. Algoritmo

```
secuencia: 10 Integer;
n,multiplo: Integer;
escribir("Escribe 10 números");
leer_secuencia(secuencia);
colocarnos_al_final(secuencia);
n←elemento_actual;
colocarnos_al_principio(secuencia);
multiplo←elemento_actual;

para x 1 .. 10 repetir
        si multiplo rem n=0 entonces
                escribir(multiplo);
        fin_si;
        avanzar(secuencia);
        multiplo←elemento_actual;
fin_repetir;
```



4. Simulación

[Para el caso de prueba que queráis, pero indicad cuál.]

Secuencia: (8,5,2,3,6,7,8,1,2,2)

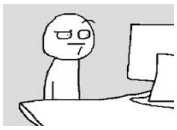
¿Entra?	N.º Vueltas	n	Múltiplo	Salida
Inicializaciones	0	2	8	-
SÍ	1	2	8	8
SÍ	2	2	5	-
SÍ	3	2	2	2
SÍ	4	2	3	-
SÍ	5	2	6	6
SÍ	6	2	7	-
SÍ	7	2	8	8
SÍ	8	2	1	-
SÍ	9	2	2	2
SÍ	10	2	2	2
NO	-	-	-	-

3º ejercicio

Dada una secuencia de 10 enteros desarrollar un algoritmo tal que nos diga si todos los números de dicha secuencia aparecen ordenados o no.

1. Especificación

Entrada: *una secuencia de 10 números enteros*
Pre: *la secuencia estará totalmente llena*
Salida: *un mensaje por pantalla*
Post: *escribir "secuencia ordenada" si "* ***todos*** *los elementos de la secuencia siguen un orden ascendente o descendente; escribir "secuencia sin orden" en caso contrario*

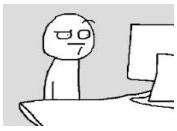


2. Casos de prueba

1 ,2 ,5, 7, 8, 9, 10 ,12, 25, 67 → Sí, sigue orden
4 ,2 ,5, 1, 8, 9, 10 ,12, 25, 67 → No, no sigue orden
1 ,2 ,5, 7, 8, 9, 10 ,12, 25, 3 → No, no sigue orden
20, 14, 12, 8, 7, 6, 4, 3, 2, 1 → Sí, sigue orden
1,2,2,2,2,2,3,3,3,4 → Sí, sigue orden
4,4,4,3,3,2,2,2,2,1 → Sí, sigue orden

3. Algoritmo

```
sec: 10 Integer;
Mm,mM: Boolean;
num1,num2: Integer;
escribir ("Escribe 10 números");
leer_secuencia(sec);
colocarnos_al_principio(sec);
ordenado· True;
num1←elemento_actual;
avanzar(sec)
num2←elemento_actual;
Mm←True;
mM←True;

repetir salir si  fuera(sec) o Mm=False y mM=False
        si num1>num2 y Mm entonces
                mM←False;
        Fin_si;
        Si num1<num2 y mM entonces
                Mm←False;
        Fin_si;
        Num1←elemento_actual;
        Avanzar(sec);
        Num2←elemento_actual;
Fin_repetir;
si mM o Mm entonces
        escribir ("Sí, sigue un orden.");
si no
        escribir ("No, no sigue un orden.")
fin_si;
```

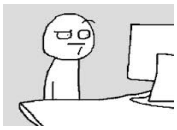


4. Simulación

[Para el caso de prueba que queráis, pero indicad cuál.]

Secuencia: (4,4,4,3,3,2,2,2,2,1)

¿Entra?	Num1	Num2	Mm (Num1>Num2)	mM (Num1<Num2)	Salida
Inicializaciones	4	4	True	True	-
SÍ	4	4	True	False	-
SÍ	4	4	True	False	-
SÍ	4	3	True	False	-
SÍ	3	3	True	False	-
SÍ	3	2	True	False	-
SÍ	2	2	True	False	-
SÍ	2	2	True	False	-
SÍ	2	2	True	False	-
SÍ	2	1	True	False	-
NO	1	FUERA SEC.	True	False	Sí, sigue un orden.



4º ejercicio

Pedir al usuario un número entero (N > 0) y dibujar un gráfico de N líneas como el siguiente:

Si N=5

```
* 0 0 0 0
* * 0 0 0
* * * 0 0
* * * * 0
* * * * *
```

1. Especificación

Entrada: Un número entero.
Pre: El número >0
Salida: Una matriz.
Post: Una matriz dividida diagonalmente por asteriscos y ceros, que tiene un área del número introducido al cuadrado.

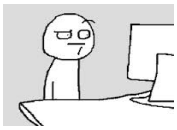
2. Casos de prueba

Entrada	Salida
1	*
4	*000 **00 ***0 ****

3. Algoritmo

[En este apartado podéis incluir el algoritmo en pseudocódigo si queréis, aunque si lo preferís podéis escribirlo directamente en Ada o en Python.]

```
n,asterisco,cero: Integer;
escribir ("Teclea un número");
leer(n);
asterisco←1;
cero←n-1;

para y 1 .. n repetir
        para x 1 .. asterisco repetir
                escribir("*");
```



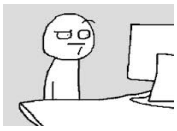

```
            para a 1 .. cero repetir
                escribir("0");
            fin_repetir;
        fin_repetir;
        escribir(salto_de_linea);
        asterisco←asterisco+1;
        cero←cero-1;
fin_repetir;
```

4. Simulación

[Para el caso de prueba que queráis, pero indicad cuál.]

Para N=5

¿Entra?	N.º Vuelta	n	Asterisco	Cero	Pantalla
Inicializaciones	-	5	1	4	-
SÍ	1	5	1	4	*0000
SÍ	2	5	2	3	**000
SÍ	3	5	3	2	***00
SÍ	4	5	4	1	****0
SÍ	5	5	5	0	*****
NO	-	-	-	-	-



5º ejercicio (Se hará en clase: no es para entregar)

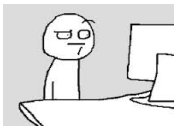
Ahora pasaremos a ejecutar nuestro primer programa en un lenguaje de programación. En este caso el programa "Hola mundo" lo ejecutaremos en ADA y en Python.

6º ejercicio

Pedir al usuario que introduzca un número entero >=0 y calcular cuántos de sus dígitos son impares.

1. Especificación

Entrada: *un número entero num*
Pre: *num >=0*
Salida: *un número entero*
Post: *valor entero >=0 que indicará el número de dígitos impares que contiene el número de entrada*



2. Casos de prueba

Entrada	Salida
123456789	5
222222222	0
111111	6
0000	0

3. Algoritmo

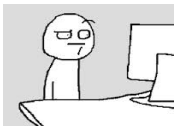
[En este apartado podéis incluir el algoritmo en pseudocódigo si queréis, aunque no es obligatorio. Lo que necesariamente deberéis hacer es incluir los archivos contar_digitos_impares.adb y contar_digitos_impares.py con el código fuente (tanto en Ada como en Python) en un fichero comprimido aparte.]

4. Simulación

[Para el caso de prueba que queráis, pero indicad cuál.]

Número= 12345

¿Entrar?	Número	Aux	Cant_impar	Pantalla
Inicializaciones	12345	0	0	-
SÍ	1234	5	1	-
SÍ	123	4	1	-
SÍ	12	3	2	-
SÍ	1	2	2	-
SÍ	0	1	3	-
NO	0	-	-	3



7º ejercicio

Pedir al usuario que introduzca un número binario (se almacenará como número entero) que comience en 1 y calcular su equivalente en decimal.

1. Especificación

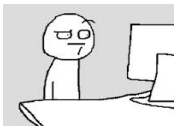
Entrada: *un número entero*
Pre: *el número será un número binario compuesto exclusivamente por 1s y 0s y comenzará por un 1*
Salida: *un número*
Post: *el número será el equivalente decimal del número de entrada, donde el peso de cada dígito binario es* $2^{posición}$.
Ejemplo $10010 = 1*2^4+0*2^3+0*2^2+1*2^1+0*2^0 = 18$
*En ADA la potencia es base**exp*
*En Python la potencia es base**exp*

2. Casos de prueba

Entrada	Salida
101	5
10000	16
11111	31
10010	18

3. Algoritmo

[En este apartado podéis incluir el algoritmo en pseudocódigo si queréis, aunque no es obligatorio. Lo que necesariamente deberéis hacer es incluir los archivos binario_a_decimal.adb y binario_a_decimal.py con el código fuente (tanto en Ada como en Python) en un fichero comprimido aparte.]



4. Simulación

[Para el caso de prueba que queráis, pero indicad cuál.]

Número= 10010

¿Entrar?	Número	Aux	Decimal	Pantalla
Inicializaciones	10010	-	0	-
SÍ	1001	0	0	-
SÍ	100	1	2	-
SÍ	10	0	0	-
SÍ	1	0	0	-
SÍ	0	1	18	-
NO	0	-	-	18