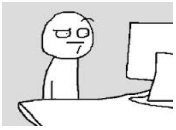
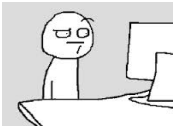


Programación Básica - Laboratorio 3
Iteraciones

Nombre: Iker Fernández y María Fernández **Fecha**: 25/09/2023

Recordemos las diferentes estructuras de control iterativas que se han presentado:

Inicialización de variables; *repetir* Acciones a repetir; Actualización de variables; *salir si* Condición*;* *fin_repetir;*	Inicialización de variables; *repetir salir si* Condición; Acciones a repetir; Actualización de variables; *fin_repetir;*
Inicialización de variables; *mientras* Condición *repetir* Acciones a repetir; Actualización de variables; *fin_repetir;*	*para* Variable Valor_inicial...Valor_final *repetir* Acciones a repetir; *fin_repetir*;



1º ejercicio

Escribir un algoritmo para sacar una ficha del EIB_Parchís de su casa. Una ficha saldrá de su casa si lanzando un dado se obtiene un cinco. De no obtener un cinco se podrá lanzar el dado sucesivas veces mientras no se superen cuatro intentos. Para desarrollar el algoritmo se podrán invocar los subprogramas *sacar_ficha()* y *lanzar_dado()*.

1) Especificación

Entrada: uno o varios número enteros, nunca más de 4.

Pre: 0 < n:valor <=6

Salida: -

Post: una ficha saldrá de casa si se obtiene un 5 con el dado en alguno de los 4 posibles intentos. Superados los intentos, imprimir mensaje: "Lo siento, límite de intentos superado"

2) Casos de prueba

Entrada: 5

Salida: ficha sale

Entrada: 2, 6, 1, 2

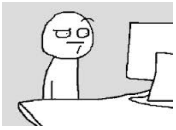
Salida: ficha no sale, y se imprime mensaje

3) Algoritmo

```
Num_dado, cont:Integer;
Escribir ("Tira el dado, con un máximo de 4 tiradas, hasta que salga 5.");
Cont ←1;
Num_dado←0;

Repetir salir si num_dado=5 o cont>4;
        Num_dado←Lanzar_dado;
        Cont ←Cont + 1;
Fin_repetir;

Si num_dado=5 entonces
        Sacar_ficha;
Fin_si;
```



2º ejercicio (Está resuelto. Solo hay que simular la 2ª opción)

Escribir un algoritmo tal que solicite al usuario un número *n* (0<*n*<=10) y escriba por pantalla la tabla de multiplicar de dicho número.

1. Especificación

Entrada: un número entero

Pre: 0 < n:valor <=10

Salida: 10 números

Post: 0 < cada valor de la tabla de multiplicar de n <=100

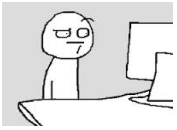
2. Casos de prueba

Entrada: 4

Salida: 4, 8, 12, 16, 20, 24, 28, 32, 36, 40

3. Algoritmo

```
multiplicador, resultado, n: Integer; --reservando memoria
multiplicador ← 1; --inicializando las variables
resultado ← 0;
leer(n);
repetir salir si multiplicador > 10; -- condición de final
        resultado ← multiplicador * n;
        escribir(resultado);
        multiplicador ← multiplicador +1; -- actualización de la variable
                                          --para llegar a que la condición sea cierta alguna vez
fin_repetir;
```



4. Simulación

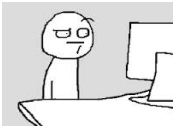
Nº de vuelta (*)	¿Entrar?	n	multiplicador	Resultado	Pantalla
Inicializaciones	-	4	1	0	
1	SI	4	2	4	4
2	SI	4	3	8	8
3	SI	4	4	12	12
4	SI	4	5	16	16
5	SI	4	6	20	20
6	SI	4	7	24	24
7	SI	4	8	28	28
8	SI	4	9	32	32
9	SI	4	10	36	36
10	SI	4	11	40	40
11	**NO**				

(*) En cada fila se representan los valores de las variables AL FINAL DE LA VUELTA.

Otro algoritmo que cumple las especificaciones es el siguiente:

```
multiplicador, resultado, n: Integer; --reservando memoria
multiplicador ← 10; --inicializando las variables
resultado ← 0;
escribir("Dame un número");
leer(n);
repetir salir si multiplicador < 1; --condición de final
    resultado ← multiplicador * n;
    escribir(resultado);
    multiplicador ← multiplicador - 1; --actualización de la variable
                                       --para llegar a que la condición sea cierta alguna vez
fin_repetir;
```

4. Simulación (para completar)



Nº de vuelta	¿Entrar?	n	multiplicador	Resultado	Pantalla
Inicializaciones	-	4	10	0	
1	SÍ	4	10	40	40
2	SÍ	4	9	36	36
3	SÍ	4	8	32	32
4	SÍ	4	7	28	28
5	SÍ	4	6	24	24
6	SÍ	4	5	20	20
7	SÍ	4	4	16	16
8	SÍ	4	3	12	12
9	SÍ	4	2	8	8
10	SÍ	4	1	4	4
11	NO				

Supongamos que la tecla “*” se ha encasquillado, y no es posible hacer uso de ella. Aun así es posible modificar el primer algoritmo, de forma que se realice la multiplicación como la ejecución de sumas sucesivas. Es decir, 2*2 es equivalente a 2+2=4, 2*3 es equivalente a 4+2=6, etcétera.

Se pide modificar el algoritmo anterior de forma que se apliquen sumas sucesivas para obtener la tabla de multiplicar.

1. Especificación

Entrada: un número entero

Pre: 0 < n:valor1 <=10

Salida: 10 números

Post: cálculo de la tabla de multiplicar por sumas sucesivas | n <= solución: 10 valores <=100

2. Casos de prueba

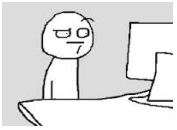
Entrada: 4

Salida: 4, 8, 12, 16, 20, 24, 28, 32, 36, 40

3. Algoritmo

contador, acumulador, n: Integer; --*reservando memoria*

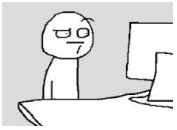
contador ← 0; --*inicializando las variables*




```
acumulador ← 0;
escribir("Dame un número");
leer(n);
repetir salir si contador =10; --condición de final
        acumulador ← acumulador + n;
        escribir(acumulador);
        contador ← contador +1; --actualización de la variable
                                --para llegar a que la condición sea cierta alguna vez
fin_repetir;
```

4. Simulación

Nº de vuelta	¿Entrar?	n	contador	acumulador	Pantalla
Inicializaciones	-	4	0	0	
1	SI	4	1	4	4
2	SI	4	2	8	8
3	SI	4	3	12	12
4	SI	4	4	16	16
5	SI	4	5	20	20
6	SI	4	6	24	24
7	SI	4	7	28	28
8	SI	4	8	32	32
9	SI	4	9	36	36
10	SI	4	10	40	40
11	**NO**				



Universidad del País Vasco Euskal Herriko Unibertsitatea

3º ejercicio

Supongamos que la tecla "*" sigue atascada y por lo tanto inutilizable. Diseñar un algoritmo que solicite dos números al usuario y calcule mediante sumas el resultado de multiplicar ambos valores.

1. Especificación

Entrada: 2 números enteros

Pre: 0 < n: valor1 y multiplicador: valor2 <=10

Salida: Un número entero

Post: 0 < resultado:valor <=100 | el valor de resultado será n*multiplicador

2. Casos de prueba (para completar)

Entrada	Salida
n = 5 y multiplicador = 1	5
n = 5 y multiplicador = 4	20
n = 10 y multiplicador = 10	100
n = 6 y multiplicador = 7	42
n = 0 y multiplicador = 8	0
n = 3 y multiplicador = 0	0
n = 1 y multiplicador = 4	4

3. Algoritmo

Num1, num2, resultado: Integer;

Escribir ("Teclea dos números.");

Leer (num1);

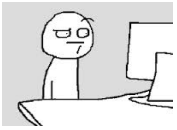
Leer (num2);

Resultado←0;

Para x 1..num2 repetir

Resultado←Resultado+num1;

Fin_repetir;



Escribir ("El resultado es" resultado);

4. Simulación (para el caso "n = 5 y multiplicador = 4")

Nº de vuelta	¿Entrar?	n	multiplicador	Resultado	Pantalla
Inicializaciones	-	5	4	0	-
1	SÍ	5	4	5	-
2	SÍ	5	4	10	-
3	SÍ	5	4	15	-
4	SÍ	5	4	20	El resultado es 20
5	NO				-

4º ejercicio

Supongamos que la tecla "/" también está atascada. Pedirle al usuario un número entero que llamaremos dividendo (0<=dividendo<=100) y otro número entero que llamaremos divisor (0<divisor<=10) y calcular el resultado de la división de ambos números y el resto, sin utilizar las operaciones / y rem: es decir, realizando la división como una repetición de restas sucesivas. Por ejemplo 6/2 = 6-2=4-2=2-2=0, como hemos realizado 3 restas significa que el resultado es 3, y el resto es 0. 7/2=7-2=5-2=3-2=1, como hemos realizado 3 restas el resultado es 3 y el resto es 1.

1. Especificación

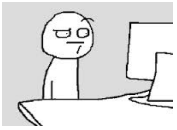
Entrada: 2 números enteros

Pre: 0 <=dividendo:valor1 <=100 y 0< divisor:valor2 <=10

Salida: 2 números enteros

Post: 0 <= cociente:valor_del_cociente_entero_de_dividendo_entre_divisor

0 <= resto:valor_del_resto_de_dividir_dividendo_entre_divisor < divisor



2. Casos de prueba

Entrada	Salida
Dividendo 6 divisor 2	Cociente 3 y resto 0
Dividendo 7 divisor 2	Cociente 3 y resto 1
Dividendo 18 divisor 4	Cociente 4 y resto 2
Dividendo 37 divisor 2	Cociente 18 y resto 1

3. Algoritmo

dividendo, divisor, cociente, resto: Integer;

-- depende de cómo lo planteéis, quizás necesitéis alguna variable más

Cociente ←0; *--inicialización (contará el número de restas que es posible hacerle al dividendo)*

Resto ← dividendo;

escribir("Dame el dividendo");

leer(dividendo);

escribir("Dame el divisor");

leer(divisor);

repetir salir si resto < divisor; *--condición de final*

resto←resto-divisor;

cociente←cociente+1; *-- actualización de la/s variables*

-- para que alguna vez la condición de la repetición llegue a ser cierta

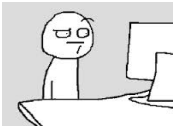
fin_repetir;

escribir("El cociente es" cociente);

escribir("El resto es" resto);

4. Simulación (para el caso que queráis)

Nº de vuelta	¿Entrar?	Dividendo	Divisor	Cociente	Resto	Pantalla
Inicializaciones	-	18	4	0	18	-
1	SÍ	18	4	1	14	-
2	SÍ	18	4	2	10	-
3	SÍ	18	4	3	6	-
4	SÍ	18	4	4	2	Cociente 4 Resto 2



5	NO	-	-	-	-	-

5º ejercicio

¡Por fin! Todas las teclas funcionan de nuevo. Ahora podemos utilizar todas las operaciones. Diseñar un algoritmo que, dado un número n >=1, escriba por pantalla todos los múltiplos pares de ese número que hay entre n y n^2.

1. Especificación

Entrada: 1 número entero

Pre: n >= 1

Salida: tantos números enteros como múltiplos pares de n haya entre n y n^2

Post: n <= múltiplos <= n^2 | son múltiplos de n y son pares

2. Casos de prueba

Entrada	Salida
2	2 4
7	14 28 42
8	8 16 24 32 40 48 56 64

3. Algoritmo

posible_multiplo, n: Integer; --*declaración*

escribir("Dame un número");

leer(n);

posible_multiplo ←n*n--*inicializar variables*

repetir salir si posible_multiplo=n*n; --*condición de fin*

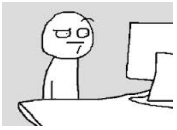
para x 1..n repetir

posible_multiplo←n*x;

si posible_multiplo rem 2=0 entonces --*condición para decidir si un posible múltiplo es realmente múltiplo par*

escribir (posible_multiplo);

fin_si;



fin_repetir;

4. Simulación

Núm. de vuelta	¿Entrar?	n	Posible_multiplo		Pantalla
inicializaciones	-	7	7		-
1	SÍ	7	7		-
2	SÍ	7	14		14
3	SÍ	7	21		-
4	SÍ	7	28		28
5	SÍ	7	35		-
6	SÍ	7	42		42
7	SÍ	7	49		-

6º ejercicio

Calcular el factorial de un número entero < 6.

1. Especificación

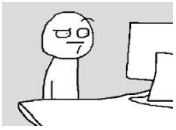
Entrada: 1 número entero

Pre: 0 <= num: valor <=6

Salida: un número entero

Post: 0 < factorial:valor <=720 | resultado= num*(num-1)! y 0!=1.

2. Casos de prueba



Entrada	Salida
0!	1
4!	24
5!	120

3. Algoritmo

factorial, num, num_aux: Integer;

-- depende de cómo lo planteéis, quizás necesitéis otra variable más (num_aux)

factorial ← 1; *--inicialización*

num_aux←num*; -- inicialización de num_aux si la utilizaseis*

escribir("Dame un número");

leer(num);

repetir salir si num_aux=0; *--condición de final*

factorial←num_aux*factorial

num_aux ← num-1; *-- actualización de la/s variables para que*

-- alguna vez la condición de la repetición llegue a ser cierta

fin_repetir;

escribir("El factorial de" num "es" factorial);

4. Simulación (para el caso que queráis)

Nº de vuelta	¿Entrar?	n	Factorial	Pantalla
Inicializaciones	-	5	1	-
1	SÍ	5	5	-
2	SÍ	5	20	-
3	SÍ	5	60	-
4	SÍ	5	120	El factorial de 5 es 120
5	NO	5	-	-

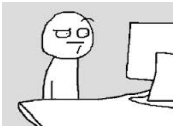
Algunas instrucciones con secuencias de datos

1) Reservar memoria para la secuencia

Por ejemplo, reservar memoria para 10 números enteros

secuencia: 10 Integer;

2) Leer desde el teclado de golpe tantos números como la secuencia tenga y depositarlos en la variable secuencia



leer_secuencia(**secuencia**);

3) Introducir un valor, guardar un valor en la posición actual de la secuencia

guardar(secuencia, n);

4) Colocarnos al principio de la secuencia

colocarnos_al_principio(secuencia);

5) Colocarnos al final de la secuencia

colocarnos_al_final(secuencia);

6) Avanzar en la secuencia

avanzar(secuencia);

así paso a paso nos podremos mover hacia adelante por la secuencia

7) Retroceder en la secuencia

retroceder(secuencia);

así paso a paso nos podremos mover hacia atrás por la secuencia

7) Obtener un elemento de la secuencia

elemento_actual(secuencia);

8) Para controlar si estamos fuera de la secuencia

fuera(secuencia); nos devolverá true cuando estemos fuera de la secuencia (de tanto avanzar – o retroceder – puede que nos suceda que nos pasemos de largo y nos salgamos de la secuencia)

7º Ejercicio

Utilizando estas operaciones, supondremos que nos mandan diseñar el algoritmo para leer una secuencia y recorrerla escribiendo por pantalla sus valores. El algoritmo podría ser el siguiente:

1. Especificación

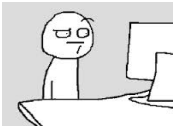
Entrada: Secuencia de 10 números enteros

Pre: -

Salida: 10 números enteros

Post: se escribirán por pantalla los valores guardados en la secuencia

2. Casos de prueba



Entrada	Salida
12000,565,7,8,2,300,4,6,9,0	12000,565,7,8,2,300,4,6,9,0

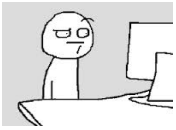
3. Algoritmo

```
secuencia1: 10 Integer; --reservamos memoria, 10 posiciones
aux: Integer;
leer_secuencia(secuencia1);
colocarnos_al_comienzo(secuencia1);
repetir salir si fuera(secuencia1);
        aux ← elemento_actual(secuencia1);
        escribir(aux);
        avanzar(secuencia1);
fin_repetir;
```

4. Simulación (para el caso de prueba dado)

Nº de vuelta	¿Entrar?	aux	Pantalla
Inicializaciones	-	-	-
1	SÍ	1200	1200
2	SÍ	565	565
3	SÍ	7	7
4	SÍ	8	8
5	SÍ	2	2
6	SÍ	300	300
7	SÍ	4	4
8	SÍ	6	6
9	SÍ	9	9
10	SÍ	0	0

8º Ejercicio



Ahora deberemos escribir un algoritmo tal que lea una secuencia de teclado y sume todos sus valores que sean pares.

1. Especificación

Entrada: una secuencia de 10 números enteros

Pre: -

Salida: 1 número entero

Post: se escribirá por pantalla el valor obtenido de calcular la suma de todos los elementos pares de la secuencia

2. Casos de prueba

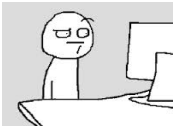
Entrada	Salida
12,565,7,8,2,300,4,6,9,0	332

3. Algoritmo

```
Secuencia: 10 Integer;
Sumatorio: Integer;
leer_secuencia(secuencia);
colocarnos_al_comienzo(secuencia);
sumatorio←0;

repetir salir si fuera (secuencia);
        si elemento_actual rem 2=0 entonces
                sumatorio←elemento_actual+sumatorio;
        fin_si;
        avanzar(secuencia);
fin_repetir;
escribir("El resultado de la suma de los números pares es" sumatorio);
```

4. Simulación (La secuencia del caso de prueba)



Nº de vuelta	¿Entrar?	Elemento_actual	Sumatorio	Pantalla
Inicializaciones	-	12	0	-
1	SÍ	12	12	-
2	SÍ	565	12	-
3	SÍ	7	12	-
4	SÍ	8	20	-
5	SÍ	2	22	-
6	SÍ	300	322	-
7	SÍ	4	326	-
8	SÍ	6	332	-
9	SÍ	9	332	-
10	SÍ	0	332	-
11	NO	-	-	El resultado de la suma de los números pares es 332