

CURSO DE PROGRAMACIÓN FULL STACK

ESTRUCTURAS DE CONTROL CON PSEINT – PARTE 2





Objetivos de la Guía

En esta guía aprenderemos a:

- Armar estructuras repetitivas.
- Usar estructuras repetitivas.
- Desarrollar la cooperación trabajando en equipo con la mesa.

GUÍA DE ESTRUCTURAS DE CONTROL – PARTE 2

¿QUÉ SON LAS ESTRUCTURAS REPETITIVAS?

Durante el proceso de creación de programas, es muy común encontrarse con que una operación o conjunto de operaciones deben repetirse muchas veces. Para ello es importante conocer las estructuras de algoritmos que permiten repetir una o varias acciones, un número determinado de veces.

Las estructuras que repiten una secuencia de instrucciones un número determinado de veces se denominan bucles, y se denomina iteración al hecho de repetir la ejecución de una secuencia de acciones.

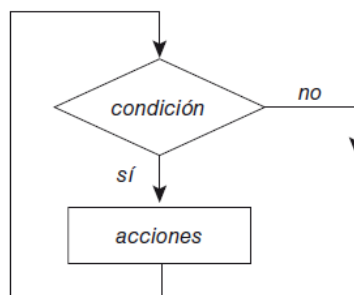
Todo bucle tiene que llevar asociada una condición, que es la que va a determinar cuándo se repite el bucle y cuando deja de repetirse.

Hay distintos tipos de bucles:

- Mientras
- Hacer Mientras
- Para

ESTRUCTURA MIENTRAS

Esta estructura repetitiva **Mientras**, es en la que el cuerpo del bucle se repite siempre que se cumpla una determinada *condición*. Cuando se ejecuta la instrucción mientras, la primera cosa que sucede es que se evalúa la condición (una expresión lógica). Si se evalúa *falsa*, no se toma ninguna acción y el programa prosigue con la siguiente instrucción. Si la expresión lógica es *verdadera*, entonces se ejecuta el cuerpo del bucle, después de lo cual se evalúa de nuevo la expresión lógica. Este proceso se repite una y otra vez mientras la expresión lógica (condición) sea verdadera, para salir del bucle la condición debe ser falsa.



Estructura Mientras en PseInt

```
Mientras expresion_logica Hacer
.....
  secuencia_de_acciones
Fin Mientras
```



Regla práctica

Las pruebas o test en las expresiones lógicas es conveniente que sean mayor o menor que en lugar de pruebas de igualdad o desigualdad. En el caso de la codificación en un lenguaje de programación, esta regla debe seguirse rígidamente en el caso de comparación de números reales, ya que como esos valores se almacenan en cantidades aproximadas las comparaciones de igualdad de valores reales normalmente plantean problemas. Siempre que realice comparaciones de números reales use las relaciones $<$, $<=$, $>$ o $>=$.



¿NECESITAS UN EJEMPLO?

```
1  Algoritmo EjemploMientras
2
3  Definir nota como entero
4
5  Escribir "Ingrese una nota valida"
6  Leer nota
7
8  /// En este bucle buscamos las notas que esten fuera de 0 o 10,
9  /// para que el bucle de verdadero y se pida la nota de nuevo.
10 /// Nosotros no estamos buscando que ingrese la nota de nuevo
11 /// cuando sea correcta, sino cuando sea incorrecta
12
13 Mientras nota < 0 o nota > 10 /// Aca ponemos una 0 porque si ponemos una Y, nunca se cumple la condición
14     /// Cuando la nota sea correcta saldrá del bucle
15     Escribir "Ingrese la nota de nuevo"
16     Leer nota
17
18 FinMientras
19
20 Escribir "La nota es correcta"
21
22 FinAlgoritmo
23
```



Pueden encontrar un ejemplo para descargar del Bucle Mientras en Aula Virtual.



MANOS A LA OBRA!

EJERCICIO VOCAL SECRETA

Diseña un programa que guarde una vocal secreta en una variable, debemos pedirle al usuario que intente adivinar la vocal secreta, e intentará tantas veces como sea necesario hasta que la adivine.

DETECCIÓN DE ERRORES

Copia y pega este código en tu programa. Deberás corregir los errores hasta lograr el siguiente resultado esperado:

```
Algoritmo Correccion_Mientras
Definir num Como Entero
//El programa ingresará números mientras sean PARES
Escribir "Ingrese un número"
Leer num
Mientras num % 2 == 0 Hacer
    Escribir "Ingrese otro número"

FinAlgoritmo
```

¿Cuál es el resultado a lograr?

```
PSelnt - Ejecutando proceso CORRECCION_MIENTRAS
*** Ejecución Iniciada. ***
Ingrese un número
> 2
Ingrese otro número
> 8
Ingrese otro número
> 10
Ingrese otro número
> 6
Ingrese otro número
> 9
```



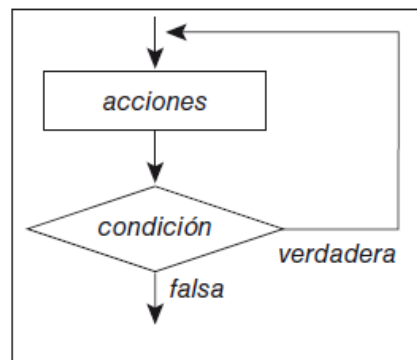
Revisemos lo aprendido hasta aquí

- Definir, implementar y diferenciar la estructura MIENTRAS. Sabiendo que es la estructura que PRIMERO valida la condición y luego ejecuta el código repetidamente MIENTRAS la condición sea verdadera.

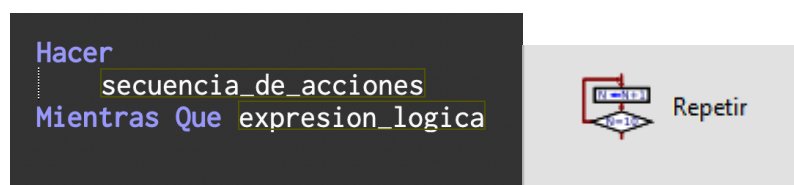
Si no pudiste interiorizar el concepto en su totalidad, **no te preocupes**, más adelante seguiremos trabajando sobre este tema.

ESTRUCTURA HACER- MIENTRAS

Esta estructura es muy similar a la anterior, sólo que a diferencia del **Mientras** el contenido del bucle **Hacer-Mientras** se ejecuta siempre al menos una vez, ya que la evaluación de la condición lógica se encuentra al final del bucle. De esta forma garantizamos que las acciones dentro de este bucle sean llevadas a cabo al menos una vez, incluso aunque la expresión lógica sea falsa.



Estructura Hacer-Mientras en PseInt:



Regla práctica

El bucle *hacer-mientras* se termina de ejecutar cuando el valor de la condición es falso. La elección entre un bucle mientras y un bucle hacer-mientras depende del problema de cómputo a resolver. En la mayoría de los casos, el bucle mientras es la elección correcta. Por ejemplo, si el bucle se utiliza para recorrer una lista de números (o una lista de cualquier tipo de objetos), la lista puede estar vacía, en cuyo caso las sentencias del bucle nunca se ejecutarán. Si se aplica un bucle hacer-mientras nos conduce a un código de errores.



¿NECESITAS UN EJEMPLO?

```
1 Algoritmo EjemploMientrasQue
2
3 Definir nota como entero
4
5
6 /// En este bucle buscamos las notas que esten fuera de 0 o 10,
7 /// para que el bucle de verdadero y se pida la nota de nuevo.
8 /// Nosotros no estamos buscando que ingrese la nota de nuevo
9 /// cuando sea correcta, sino cuando sea incorrecta
10 Hacer
11     /// A diferencia del mientras pedimos la nota adentro,
12     /// ya que se el bucle se corre por lo menos una vez
13     Escribir "Ingrese una nota valida"
14     Leer nota
15
16 Mientras Que nota < 0 o nota > 10
17     /// Ponemos la condicion al final
18
19
20     Escribir "La nota es correcta"
21
22 FinAlgoritmo
23
```

```
PSelnt - Ejecutando proceso EJEMPLOMIENTRAQUE
*** Ejecución Iniciada. ***
Ingrese una nota valida
> 12
Ingrese una nota valida
> 12
Ingrese una nota valida
> 15
Ingrese una nota valida
> 6
La nota es correcta
*** Ejecución Finalizada. ***
```



Pueden encontrar un ejemplo para descargar del Bucle Hacer-Mientras en Aula Virtual.



MANOS A LA OBRA!

EJERCICIO VOCAL SECRETA – PARTE 2

Vamos a hacer nuevamente el ejercicio de la vocal misteriosa, pero esta vez con una estructura Hacer-Mientras. ¿Puedes notar cual es la diferencia entre ambas estructuras?

DETECCIÓN DE ERRORES

Copia y pega este código en tu programa. Deberás corregir los errores hasta lograr el siguiente resultado esperado:

Algoritmo Correccion_HacerMientras

num Como Entero

Repet

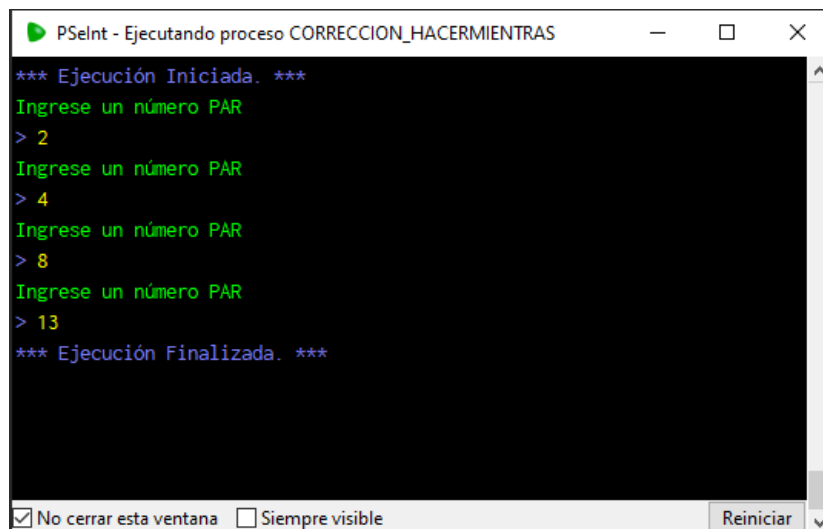
 Escribir "Ingrese un número PAR"

 Leer num

Mientras Qe num MOD 2 = 0

FinAlgoritmo

¿Cuál es el resultado a lograr?



```
*** Ejecución Iniciada. ***
Ingrese un número PAR
> 2
Ingrese un número PAR
> 4
Ingrese un número PAR
> 8
Ingrese un número PAR
> 13
*** Ejecución Finalizada. ***
```



Revisemos lo aprendido hasta aquí

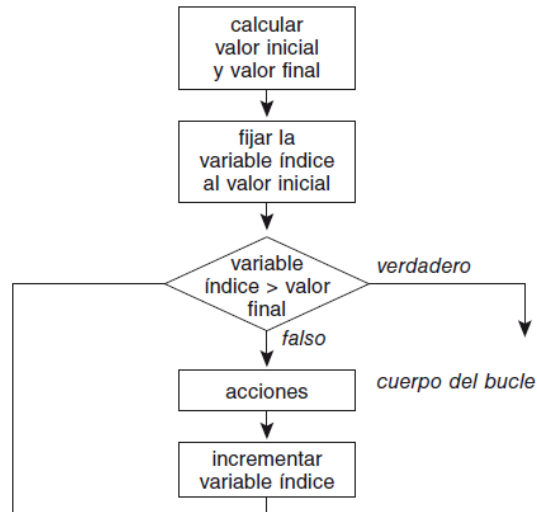
- Identificar, diferenciar y elaborar una estructura HACER MIENTRAS, cuya primera validación de la condición se da DESPUÉS de la primera ejecución del bloque de código. Es decir que esta estructura siempre se ejecutará AL MENOS UNA VEZ.

Si no pudiste interiorizar el concepto en su totalidad, **no te preocupes**, más adelante seguiremos trabajando sobre este tema.

ESTRUCTURA PARA

La estructura **Para** es un poco más compleja que las anteriores y nos permite ejecutar un conjunto de acciones, para cada paso de un conjunto de elementos. Su implementación depende del lenguaje de programación, pero en términos generales podemos identificar tres componentes: la *inicialización*, *finalización* y el *incremento*.

La estructura **Para** comienza con un valor inicial de una variable llamada índice y las acciones especificadas se ejecutan x cantidad de veces, hasta que el valor índice llegue al valor final, *a menos que el valor inicial sea mayor que el valor final*. La variable índice se incrementa en uno y si este nuevo valor no excede al final, se ejecutan de nuevo las acciones. Por consiguiente, las acciones específicas en el bucle se ejecutan para cada valor de la variable índice desde el valor inicial hasta el valor final con el incremento de uno en uno.



Estructura Para en Pselnt:

```

Para variable_numerica<-valor_inicial Hasta valor_final Con Paso paso Hacer
.....
  secuencia_de_acciones
Fin Para
  
```



El incremento de la variable índice (variable_numerica) siempre es 1 si no se indica expresamente lo contrario en el valor de *con paso*. Dependiendo del tipo de lenguaje, es posible que el incremento sea distinto de uno, positivo o negativo. La variable índice o de control (variable_numerica) normalmente será de tipo entero y es normal emplear como nombres las letras i, j, k.

Si el valor_inicial de la variable índice es menor que el valor_final, los incrementos, es decir los pasos, deben ser positivos, ya que en caso contrario la secuencia de acciones no se ejecutaría. De igual modo, si el valor_iniciales es mayor que el valor_final, el paso debe ser en este caso negativo, es decir, decremento.



<pre> 1 Algoritmo EjemploPara 2 3 Definir i Como Entero 4 5 Para i <- 1 Hasta 10 Con Paso 1 Hacer 6 7 Escribir "La tabla del 2 es:" i * 2 8 9 Fin Para 10 11 12 FinAlgoritmo 13 </pre>	<pre> PSELnt - Ejecutando proceso EJEMPLOPARA *** Ejecución Iniciada. *** La tabla del 2 es:2 La tabla del 2 es:4 La tabla del 2 es:6 La tabla del 2 es:8 La tabla del 2 es:10 La tabla del 2 es:12 La tabla del 2 es:14 La tabla del 2 es:16 La tabla del 2 es:18 La tabla del 2 es:20 *** Ejecución Finalizada. *** </pre>
--	--



Pueden encontrar un ejemplo para descargar del Bucle Para en el Aula Virtual.



MANOS A LA OBRA!

EJERCICIO NUMERO MAYOR

Escribir una estructura PARA que le solicite al usuario varios números y al finalizar muestre el mayor número ingresado.

DETECCIÓN DE ERRORES

Algoritmo correccion_Para

Para <-0 Hasta Con Paso Hacer

 Escribir "Imprimimos el valor de i"

 Escribir i

Fin Para

FinAlgoritmo

¿Cuál es el resultado a lograr?

```
PSeInt - Ejecutando proceso SIN_TITULO
Imprimimos el valor de i
2
Imprimimos el valor de i
4
Imprimimos el valor de i
6
Imprimimos el valor de i
8
Imprimimos el valor de i
10
Imprimimos el valor de i
12
*** Ejecución Finalizada. ***
 No cerrar esta ventana  Siempre visible Reiniciar
```



Revisemos lo aprendido hasta aquí

- Identificar, construir y utilizar una estructura **PARA**. En esta estructura tenemos el control absoluto de las repeticiones, ya que podemos determinar la inicialización, límite y aumento del valor de *i*.

Si no pudiste interiorizar el concepto en su totalidad, **no te preocupes**, más adelante seguiremos trabajando sobre este tema.

FUNCIONES PSEINT

Además de empezar a implementar las estructuras de control, vamos a empezar a utilizar las **funciones de Pselnt**. Las funciones, son herramientas que nos proporciona Pselnt y cumplen el propósito de *ayudarnos a resolver ciertos problemas*. Supongamos que tenemos que calcular la raíz cuadrada de un número, Pselnt cuenta con una función que pasándole un número, nos devuelve el resultado de su raíz cuadrada. Ese resultado que devuelve, se lo podemos asignar a una variable o lo podemos concatenar con un escribir para mostrar el resultado sin la necesidad de una variable.

También, las *funciones se pueden utilizar dentro de cualquier expresión, de cualquier estructura, y cuando se evalúe la misma, se reemplazará por el resultado correspondiente.*

Tenemos dos tipos de funciones, funciones matemáticas y funciones de cadenas de texto. Las funciones matemáticas, reciben un sólo parámetro de tipo numérico y devolverán un solo valor de tipo numérico. Las funciones de cadenas, en cambio, reciben un solo parámetro de tipo cadena, pero pueden devolver un valor de tipo cadena o de tipo numérico según la función que se use.

Funciones	Significado
RC(número)	Devuelve la raíz cuadrada del número.
ABS(número)	Devuelve el valor absoluto del número
LN(número)	Devuelve el logaritmo natural del número
EXP(número)	Devuelve la función exponencial del número.
SEN(número)	Devuelve el seno de número.
COS(número)	Devuelve el coseno de número.
TAN(número)	Devuelve la tangente de número.
ASEN(número)	Devuelve el arcoseno de número.

ACOS(número)	Arcocoseno de x
ATAN(número)	Arcotangente de x
MOD	Devuelve el módulo (resto de la división entera).
TRUNC(número)	Trunca el valor x (parte entera de x)
REDOND(número)	Redondea al valor más cercano a x
AZAR(número)	Entero aleatorio entre 0 y x -1
ALEATORIO(min,max)	Entero aleatorio entre valor mínimo y máximo



¿NECESITAS UN EJEMPLO?

Escribir "Raíz cuadrada de 9: " `rc(9)`

Escribir "Resto de 4/2: " `4 MOD 2`

Escribir "Valor absoluto de -3: " `abs(-3)`

Escribir "Seno de 90 grados: " `sen(90 * PI / 180)`

Escribir "Truncamos 3.7: " `trunc(3.7)`

Escribir "Redondeamos 2.7: " `redon(2.7)`

Escribir "Un número al azar del 0 al 9: " `azar(10)`

Escribir "Un número al azar entre 10 y 20: " `aleatorio(10,20)`

Del código anterior los resultados serían:

Raíz cuadrada de 9: **3**

Resto e 4/2: **0**

Valor absoluto de -3: **3**

Seno de 90 grados: **1**

Truncamos 3.7: **3**

Redondeamos 2.7: **3**

Un número al azar del 0 al 9: **6**

Un número al azar entre 10 y 20: **14**



Pueden encontrar un ejemplo para descargar de Funciones Matemáticas en el Aula Virtual.

FUNCIONES CADENAS DE TEXTO

Algunas funciones de cadenas de texto utilizan las posiciones de cada letra de una cadena. Esto significa que, si tengo la palabra Hola, la cadena tendrá 4 posiciones, en PselInt las posiciones de las letras arrancan en 0. Entonces para la cadena Hola, nuestras posiciones serían: 0: **H**, 1: **o**, 2: **l** y 3: **a**.

Funciones	Significado
Longitud(cadena)	Devuelve la cantidad de letras que compone la cadena.
Mayusculas(cadena)	Devuelve una copia de la cadena con todas sus letras en mayúsculas.
Minusculas(cadena)	Devuelve una copia de la cadena con todas sus letras en minúsculas.
Subcadena(cadena, posición_inicial, posición_final)	Devuelve una nueva cadena que consiste en la parte de la cadena que va desde la posición pos_inicial hasta la posición pos_final.
Concatenar(cadena, cadena2)	Devuelve una nueva cadena que resulta de unir las cadenas cadena1 y cadena2.
ConvertirANumero(cadena)	Recibe una cadena compuesta de números y devuelve la cadena como una variable numérica.
ConvertirACadena(cadena)	Recibe un número y devuelve una variable cadena de caracteres de dicho número.



¿NECESITAS UN EJEMPLO?

Definir cadena1,cadena2 como cadena

```
cadena1 = "programacion"
```

```
cadena2 = "EGG"
```

Escribir "La longitud de cadena1 es: " longitud(cadena1)

Escribir "El primer carácter de cadena1 es: " subcadena(cadena1,0,0)

Escribir "La cadena1 en mayúsculas es: " mayusculas(cadena1)

Escribir "La cadena2 en minusculas es: " minusculas(cadena2)

Escribir "La cadena concatenada queda como: " concatenar(cadena1," es muy interesante")

Escribir “La cadena convertida a numero queda:” `convertirANumero("10")`

Del código anterior los resultados serían:

La longitud de cadena1 es: 12

El primer carácter de cadena1 es: p

La cadena1 en mayúsculas es: PROGRAMACION

La cadena2 en minúsculas es: egg

La cadena concatenada queda como: programacion es muy interesante

La cadena convertida a numero queda: 10



Pueden encontrar un ejemplo para descargar de Funciones de Cadenas de Texto en el Aula Virtual.

EJERCICIOS DE APRENDIZAJE

Llegamos hasta acá y adquirimos diversas capacidades para controlar la ejecución de nuestro programa en función de estructuras repetitivas, o bucles, que nosotros mismos determinamos. Es hora de poner en práctica lo aprendido, comparar con tus compañeros de mesa cómo lo resolvió cada uno y conversar entre ustedes aquellos conceptos que no hayan quedado del todo claros. Recuerda que, si no logras resolver un ejercicio, debes avisarle al Facilitador de tu mesa para que te ayude, o haga que el resto de la mesa trabaje en conjunto para explicarte. Si aún con la ayuda, no pueden resolverlo, llamen a un Mentor con el botón de ayuda.



VIDEOS: Te sugerimos ver los videos relacionados con este tema, antes de empezar los ejercicios, los podrás encontrar en tu aula virtual o en nuestro canal de YouTube.

Para cada uno de los siguientes ejercicios realizar el análisis del problema e indicar cuáles son los datos de entrada y cuáles son los datos de salida. Escribir luego el programa en PseInt.

Bucle “Mientras”

1. Escriba un programa que valide si una nota está entre 0 y 10, sino está entre 0 y 10 la nota se pedirá de nuevo hasta que la nota sea correcta.
2. Escriba un programa en el cual se ingrese un valor límite positivo, y a continuación solicite números al usuario hasta que la suma de los números introducidos supere el límite inicial.
3. Dada una secuencia de números ingresados por teclado que finaliza con un -1, por ejemplo: 5,3,0,2,4,4,0,0,2,3,6,0,.....,-1; realizar un programa que calcule el promedio de los números ingresados. Suponemos que el usuario no insertará número negativos.

Bucle “Hacer – Mientras Que”

4. Teniendo en cuenta que la clave es “eureka”, escribir un programa que nos pida ingresar una clave. Sólo se cuenta con 3 intentos para acertar, si fallamos los 3 intentos se deberá mostrar un mensaje indicándonos que hemos agotado esos 3 intentos. Si acertamos la clave se deberá mostrar un mensaje que indique que se ha ingresado al sistema correctamente.
5. Escribir un programa que lea números enteros hasta teclear 0 (cero). Al finalizar el programa se debe mostrar el máximo número ingresado, el mínimo, y el promedio de todos ellos.

Bucle “Para”

6. Escribir un programa que calcule el cuadrado de los 9 primeros números naturales e imprima por pantalla el número seguido de su cuadrado. Ejemplo: “2 elevado al cuadrado es igual a 4”, y así sucesivamente.
7. Realizar un programa que pida una frase y el programa deberá mostrar la frase con un espacio entre cada letra. La frase se mostrará así: H o l a. Nota: recordar el funcionamiento de la función Subcadena().

NOTA: En PselInt, si queremos escribir sin que haya saltos de línea, al final de la operación “**escribir**” escribimos “**sin saltar**”. Por ejemplo:

Escribir sin saltar “Hola, “

Escribir sin saltar “cómo estás?”

Imprimirá por pantalla: Hola, cómo estás?

8. Un docente de Programación tiene un listado de 3 notas registradas por cada uno de sus N estudiantes. La nota final se compone de un trabajo práctico Integrador (35%), una Exposición (25%) y un Parcial (40%). El docente requiere los siguientes informes claves de sus estudiantes:

- Nota promedio final de los estudiantes que reprobaron el curso. Un estudiante reprueba el curso si tiene una nota final inferior a 6.5
- Porcentaje de alumnos que tienen una nota de integrador mayor a 7.5.
- La mayor nota obtenida en las exposiciones.
- Total de estudiantes que obtuvieron en el Parcial entre 4.0 y 7.5.

El programa pedirá la cantidad de alumnos que tiene el docente y en cada alumno pedirá las 3 notas y calculará todos informes claves que requiere el docente.

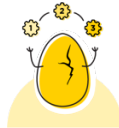
Bucles Anidados

9. Realizar un programa que lea un número entero (tamaño del lado) y a partir de él cree un cuadrado de asteriscos de ese tamaño. Los asteriscos sólo se verán en el borde del cuadrado, no en el interior. Por ejemplo, si se ingresa el número 4 se debe mostrar:

```
* * * *
*   *
*   *
* * * *
```

Nota: Recordar el uso del escribir sin saltar en PselInt.

10. Una compañía de seguros tiene contratados a n vendedores. Cada vendedor realiza múltiples ventas a la semana. La política de pagos de la compañía es que cada vendedor recibe un sueldo base más un 10% extra por comisiones de sus ventas. El gerente de la compañía desea saber, por un lado, cuánto dinero deberá pagar en la semana a cada vendedor por concepto de comisiones de las ventas realizadas, y por otro lado, cuánto deberá pagar a cada vendedor como sueldo total (sueldo base + comisiones). Para cada vendedor ingresar cuanto es su sueldo base, cuantas ventas realizó y cuanto cobró por cada venta.



¿Lograste los objetivos de la guía?

¡Terminamos los ejercicios fundamentales de esta guía! **Te invitamos a ir a tu aula virtual y contestar una breve encuesta para revisar los objetivos de la esta guía**, reflexiona sobre cada uno de ellos haciéndote la siguiente pregunta ¿Pude aplicar este objetivo al realizar los ejercicios? Si es así, ¡Excelente!

¿Terminaste con los ejercicios fundamentales? hay más ejercicios para que puedas seguir practicando.

¿No pudiste cumplir los objetivos de la guía?

Si aún no has logrado tildar los objetivos propuestos, **no te preocupes**, a través de Slack contacta a un Coach para que te ayude a diseñar una estrategia personalizada para entender qué pasó y poder avanzar.

EJERCICIOS DE APRENDIZAJE EXTRA

Estos van a ser ejercicios para reforzar los conocimientos previamente vistos. Estos pueden realizarse cuando hayas terminado la guía y tengas una buena base sobre lo que venimos trabajando. Además, si ya terminaste la guía y te queda tiempo libre en las mesas, puedes continuar con estos ejercicios extra, recordando siempre que no es necesario que los termines para continuar con el tema siguiente. Por último, recordá que la prioridad es ayudar a los compañeros de la mesa y que cuando tengas que ayudar, lo más valioso es que puedas explicar el ejercicio con la intención de que tu compañero lo comprenda, y no sólo mostrarlo. ¡Muchas gracias!

Bucle “Mientras”

1. Escriba un programa en el cual se ingrese un número y mientras ese número sea mayor de 10, se pedirá el número de nuevo.
2. Escriba un programa que solicite dos números enteros (mínimo y máximo). A continuación, se debe pedir al usuario que ingrese números enteros situados entre el máximo y mínimo. Cada vez que un número se encuentre entre ese intervalo, se sumara uno a una variable. El programa terminará cuando se escriba un número que no pertenezca a ese intervalo, y al finalizar se debe mostrar por pantalla la cantidad de números ingresados dentro del intervalo.

3. Escriba un programa que solicite al usuario números decimales mientras que el usuario escriba números mayores al primero que se ingresó. Por ejemplo: si el usuario ingresa como primer número un 3.1, y luego ingresa un 4, el programa debe solicitar un tercer número. El programa continuará solicitando valores sucesivamente mientras los valores ingresados sean mayores que 3.1, caso contrario, el programa finaliza.
4. Calcular las calificaciones de un grupo de alumnos. La nota final de cada alumno se calcula según el siguiente criterio: la parte práctica vale el 10%; la parte de problemas vale el 50% y la parte teórica el 40%. El programa leerá el nombre del alumno, las tres notas obtenidas, mostrará el resultado por pantalla, y a continuación volverá a pedir los datos del siguiente alumno hasta que el nombre sea una cadena vacía. Las notas deben estar comprendidas entre 0 y 10, y si no están dentro de ese rango no se imprimirá el promedio y se mostrará un mensaje de error.
5. Escribir un programa que calcule cuántos dígitos tiene un número entero positivo sin convertirlo a cadena (pista: se puede hacer dividiendo varias veces entre 10). Nota: investigar la función `trunc()`.

Bucle “Hacer – Mientras Que”

6. Realizar un programa que solicite al usuario su código de usuario (un número entero mayor que cero) y su contraseña numérica (otro número entero positivo). El programa no le debe permitir continuar hasta que introduzca como código 1024 y como contraseña 4567. El programa finaliza cuando ingresa los datos correctos.
7. Se debe realizar un programa que:
 - 1º) Pida por teclado un número (entero positivo).
 - 2º) Pregunte al usuario si desea introducir o no otro número.
 - 3º) Repita los pasos 1º y 2º mientras que el usuario no responda n/N (no).
 - 4º) Muestre por pantalla la suma de los números introducidos por el usuario.
8. Hacer un algoritmo para calcular la media de los números pares e impares, sólo se ingresará diez números.
9. Se pide escribir un programa que calcule la suma de los N primeros números pares. Es decir, si ingresamos el número 5 como valor de N, el algoritmo nos debe realizar la suma de los siguientes valores: $2+4+6+8+10$.
10. Programar un juego donde la computadora elige un número al azar entre 1 y 10, y a continuación el jugador tiene que adivinarlo. La estructura del programa es la siguiente:
 - 1º) El programa elige al azar un número n entre 1 y 10.
 - 2º) El usuario ingresa un número x.
 - 3º) Si x no es el número exacto, el programa indica si n es más grande o más pequeño que el número ingresado.
 - 4º) Repetimos desde 2) hasta que x sea igual a n.

El programa tiene que imprimir los mensajes adecuados para informarle al usuario qué hacer y qué pasó hasta que adivine el número.

NOTA: Para generar un número aleatorio entre 1 y 10 se puede utilizar la función `Aleatorio(limite_inferior, limite_superior)` de `Pselnt`.

Bucle “Para”

11. Realizar un programa que muestre la cantidad de números que son múltiplos de 2 o de 3 comprendidos entre 1 y 100.
12. Escribir un programa que calcule la suma de los N primeros números naturales. El valor de N se leerá por teclado.
13. Siguiendo el ejercicio 20 de los ejercicios principales, ahora deberemos hacer lo mismo pero que la cadena se muestre al revés. Por ejemplo, si tenemos la cadena: Hola, deberemos mostrar **a l o H**.

Bucles Anidados

14. Escriba un programa que lea un número entero (altura) y a partir de él cree una escalera invertida de asteriscos con esa altura. Por ejemplo, si ingresamos una altura de 5 se deberá mostrar:

```
*****  
****  
***  
**  
*
```

15. La función factorial se aplica a números enteros positivos. El factorial de un número entero positivo ($n!$) es igual al producto de los enteros positivos desde 1 hasta n:

$$n! = 1 * 2 * 3 * 4 * 5 * (n-1) * n$$

Escriba un programa que calcule las factoriales de todos los números enteros desde el 1 hasta el 5. El programa deberá mostrar la siguiente salida:

$$!1 = 1$$

$$!2 = 1*2 = 2$$

...

$$!5 = 1*2*3*4*5 = 120$$