

Tarea: Técnicas de Programación y pseudocódigo

Captulo 3: BUCLES



CFPI: A Carballeira

Módulo: Programación

Contenido

Estructuras Repetitivas.	3
Ciclo Mientras (while)	3
Buenas prácticas del uso de MIENTRAS	4
Ejercicios bucle Mientras (while)	7
Ciclo Repetir - Mientras (do- while)	7
Buenas prácticas en el uso del bucle REPETIR (do ... while)	8
Ejercicios bucle repetir	9
Bucle PARA (for)	9
Buenas prácticas en el uso del bucle PARA.....	10
Ejercicios bucle PARA (FOR)	11
CONCLUSIONES	12
Ejercicios de bucles:.....	13
Bucles con números Aleatorios	16
Números aleatorios	16
Juego Tirar Dados	17
Juego de Generala	17
Juego: Precio del Petróleo.....	17
Tiro de penalties	18

Diagramas de flujo y pseudocódigo

Estructuras Repetitivas.

Ciclo Mientras (while)

La instrucción Mientras ejecuta una secuencia de instrucciones seguidas del comando Hacer mientras una condición sea verdadera. Al ejecutarse esta instrucción, primero se evalúa la condición. Si la condición resulta verdadera, se ejecuta una vez la secuencia de instrucciones que forman el cuerpo del ciclo.

Luego se vuelve a evaluar la condición y, si es verdadera, la ejecución se repite. Caso contrario de que sea falsa, sale del ciclo y el programa continúa abajo del comando FinMientras.

Tener en cuenta que **las instrucciones del cuerpo del ciclo pueden no ejecutarse nunca, si al evaluar por primera vez la condición resulta ser falsa**. Por otro lado, **si la condición siempre es verdadera, al ejecutar esta instrucción se produce un ciclo infinito**. Aún de evitarlo, las instrucciones del cuerpo del bucle deben contener alguna instrucción que modifique la o las variables involucradas en la condición, de modo que ésta sea falsada en algún momento y así finalice la ejecución del ciclo.



Proceso while

Definir A, L como ENTERO;

L=0;

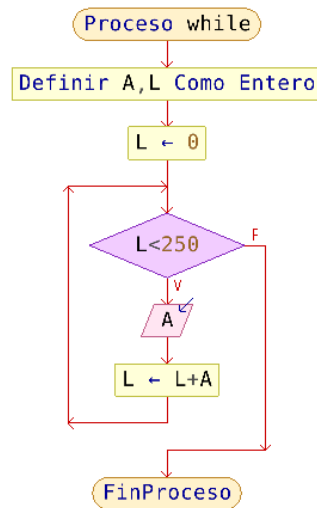
Mientras L<250 Hacer

Leer A;

L<-L+A;

FinMientras

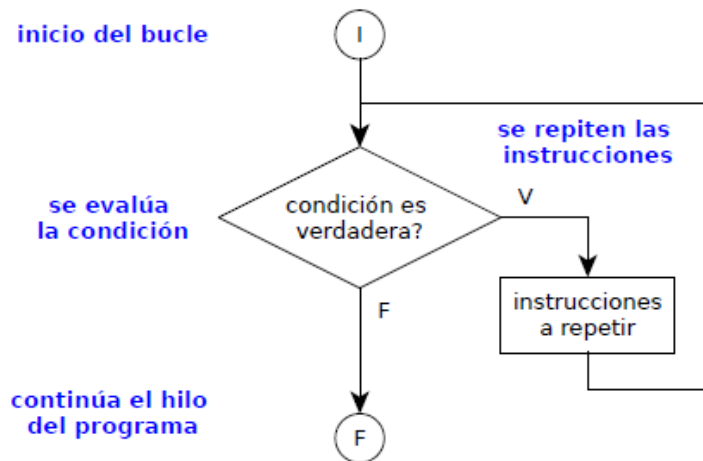
FinProceso



Buenas prácticas del uso de MIENTRAS

Esta estructura requiere que se satisfaga la condición para luego ejecutar el bloque de instrucciones a repetir.

Es una forma usada para cumplir una condición antes de pasar al siguiente bloque. Si usamos la condición del punto anterior, se usará la negación de la expresión, pues para continuar al siguiente bloque se usará el lado falso.



Ejemplo - Validación de datos de entrada con Mientras-FinMientras

En este ejemplo se muestra como validar datos de entrada en el rango 0 a 10.

Proceso while3

Definir nota **Como Entero**;

Leer nota;

Mientras (nota<0 **O** nota>10) **Hacer**

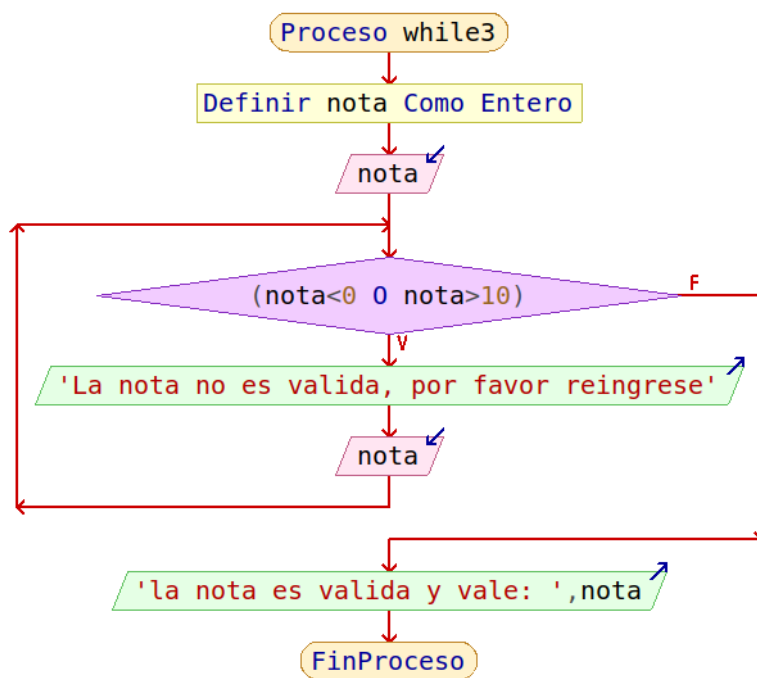
Escribir "La nota no es valida, por favor reingrese";

Leer nota;

FinMientras

Escribir "la nota es valida y vale: ",nota;

FinProceso



Ejemplo – Acumuladores con Mientras-FinMientras

Un ejemplo del uso de acumuladores es el siguiente ejemplo, un programa capaz de sumar muchos números, todos los que el usuario quisiera, y en el momento que introduzca "0" debe mostrarse el resultado de la suma, podría ser así:

Algoritmo Mientras01

```
Escribir "Dime un numero";  
Leer x;  
suma <- 0; //variable acumulador  
Mientras x <> 0 Hacer  
    suma <- suma + x;  
    Escribir "Hasta ahora, la suma es ", suma;  
    Escribir "Dime otro numero";  
    Leer x;  
FinMientras  
Escribir "Terminado";
```

FinAlgoritmo

Ejemplo – Acumuladores con Mientras-FinMientras

Supongamos que deseamos que el usuario introduzca todos los números que quiera y se muestre la media. Necesitamos llevar un conteo de los números insertados.

Algoritmo Mientras02

```
Escribir "Dime un numero";  
Leer x;  
suma <- 0; //variable acumulador  
n_numero <- 0; //variable contadora  
Mientras x <> 0 Hacer  
    suma <- suma + x;  
    n_numeros=n_numeros + 1;  
    Escribir "Hasta ahora, la suma es ", suma;  
    Escribir "Dime otro numero";
```

```
Leer x;  
  
FinMientras  
  
Escribir "Terminado";  
  
FinAlgoritmo
```

Ejercicios bucle Mientras (while)

1. Crea un programa que pida al usuario una contraseña, de forma repetitiva mientras que no introduzca "1234". Cuando finalmente escriba la contraseña correcta, se le dirá "Bienvenido" y terminará el programa.
2. Haz un programa que permita calcular la suma de pares de números. Pedirá dos números al usuario y mostrará su suma, volviendo a repetir hasta que ambos números introducidos sean 0.
3. Crea un programa que genere dos números al azar entre el 0 y el 100, y pida al usuario que calcule e introduzca su suma. Si la respuesta no es correcta, deberá volver a pedirla tantas veces como sea necesario hasta que el usuario acierte.

Pista: para generar un número al azar del 0 al 100 puedes hacer `numero <- AZAR(101)`

Ciclo Repetir - Mientras (do-while)

La sentencia Repetir entra a un bucle, donde se ejecuta las instrucciones que le proceden, y luego evalúa una condición determinada por el comando Mientras Que <condición>; si esta condición es verdadera el ciclo sigue ejecutándose, sale del mismo una vez que la condición evaluada es falsa.

Se debe tener en cuenta que el cuerpo del ciclo se ejecuta al menos una vez, ya que primero ejecuta las instrucciones y luego evalúa la condición de n. Al igual que el Lazo Mientras, hay que modificar en el cuerpo del bucle la o las variables que intervienen en la condición, de manera tal que en algún momento obtengamos un falso, y no quedemos en un bucle infinito.

Veamos un ejemplo:

Proceso do_while

```
Definir suma,x Como Entero;
```

```
suma = 0;
```

```
Repetir
```

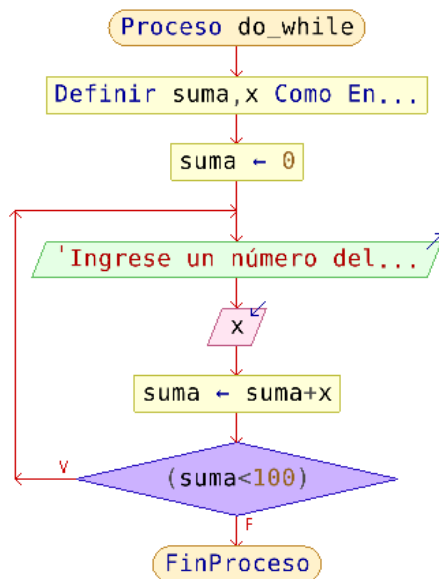
```
    Escribir "Ingrese un número del acumulador: ";
```

```
    Leer x;
```

```
suma=suma +x;
```

Mientras Que (suma<100)

FinProceso



Buenas prácticas en el uso del bucle REPETIR (do ... while)

Es también muy frecuente que un bloque de programa que quizá se repita, deba ejecutarse al menos una vez. Por ejemplo, si queremos pedir un dato al usuario, quizá exista algún error y haya que insistir, pero al menos deberemos pedírselo una primera vez.

En estos casos, la estructura "mientras" no es la más adecuada: no podemos comprobar la condición al principio, sino después de haber pedido el valor. En estos casos (que son muy frecuentes), sería más razonable usar otra estructura de programación en la que la condición se compruebe después de dar ciertos pasos. Esa estructura es "repetir... hasta":

Por ejemplo, un programa que pida al usuario una clave de acceso, y que no le permita seguir hasta que la introduzca correctamente, se podría hacer así:

Algoritmo Repetir01

Repetir

Escribir "Dime tu clave de acceso";

Leer clave;

Si clave <> 1234 **Entonces**

Escribir "Clave incorrecta";


```
FinSi  
  
Hasta Que clave=1234  
  
Escribir "Bienvenido!";  
  
FinAlgoritmo
```

Este bucle también se emplea con variables CONTADORAS y ACUMULADORAS:

Ejercicios bucle repetir

1. Crea un programa que pida al usuario un código de usuario y una contraseña. Deberá repetirse hasta que el código sea "1" y la contraseña sea "1234".
2. Haz un programa que permita calcular la suma de pares de números. Pedirá dos números al usuario y mostrará su suma, volviendo a repetir hasta que ambos números introducidos sean 0. Esta vez deberás usar "Repetir", por lo que tu solución no será igual que la del ejercicio 6.2, que empleaba "Mientras".
3. Prepara un programa que divida dos números que introduzca el usuario. Si el segundo número es cero, se le deberá avisar y volver a pedir tantas veces como sea necesario, hasta que introduzca un número distinto de cero, momento en que se calculará y mostrará el resultado de la división

Bucle PARA (for)

La instrucción Para ejecuta una secuencia de instrucciones un número determinado de veces, denido por el valor seguido del comando Hasta.

Al ingresar al bloque, la variable de control recibe el valor inicial y se ejecuta la secuencia de instrucciones que forma el cuerpo bucle. Luego se incrementa la variable de control en un paso de unidades determinado y se evalúa la condición de n. Se repite el ciclo hasta que de falso la condición.

Si se omite la cláusula Con Paso <paso>, la variable de control se incrementará en 1.

Ejemplo:

Definir i como ENTERO;

Escribir “Los primeros números pares son: “;

Para i<-0 Hasta 10 Con Paso 2 Hacer

Escribir i, “; “;

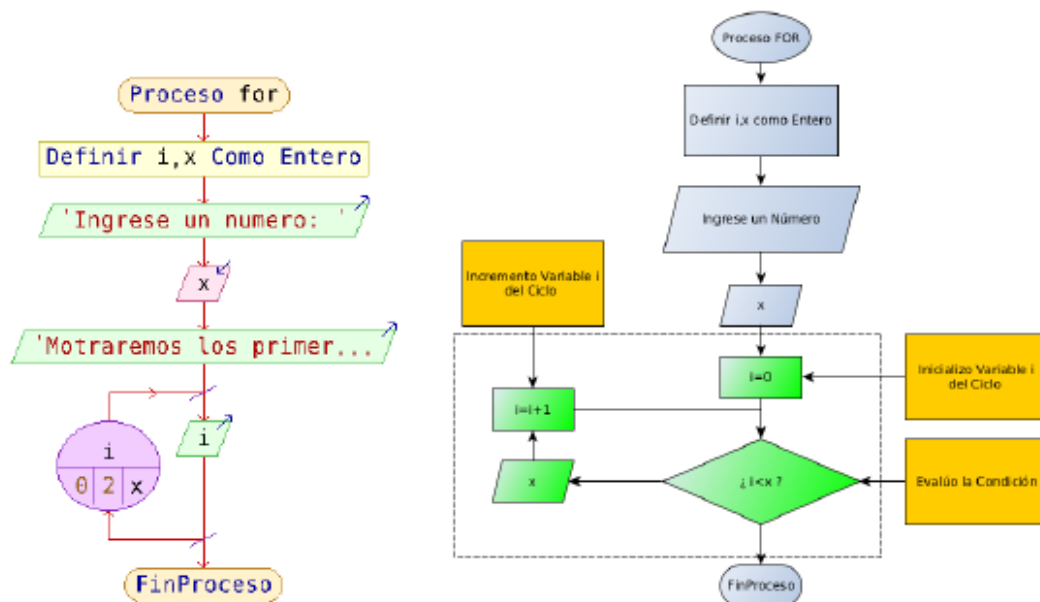
FinPara

Proceso for

```
Definir i , x como ENTERO;  
Escribir "Ingrese un numero: ";  
Leer x;  
Escribir "Motraremos los primeros pares menores a ",x ;  
Para i=0 Hasta x Con Paso 2 Hacer  
    Escribir i;
```

FinPara

FinProceso



Notar que NO necesariamente debe comenzar en Cero.

Notar que NO necesariamente el incremento o paso debe ser de 2.

Buenas prácticas en el uso del bucle PARA

En muchas ocasiones, no queremos que algo se repita mientras se cumpla una condición, sino un cierto número de veces. Por ejemplo, para escribir "Hola" 3 veces en pantalla existe una orden más cómoda que la orden "mientras" o la orden "repetir... hasta". Es la orden "para", que hace que una variable tome una serie de valores que se van incrementando.

Por ejemplo, un programa que mostrara los números del 1 al 10, podría ser:

```
Algoritmo Para01  
  
    Para x <- 1 Hasta 10 Hacer  
        Escribir x;  
    FinPara  
FinAlgoritmo
```

Si no queremos avanzar de uno en uno, sino con un incremento distinto, podemos indicar otro tamaño de "paso":

```
Algoritmo Para02  
  
    Para x <- 10 Hasta 20 Con Paso 2 Hacer  
        Escribir x;  
    FinPara  
FinAlgoritmo
```

Pero para lo que realmente se emplea este bucle de repetición es para recorrer una estructura de datos llamada MATRIZ por lo que de momento no lo emplearemos mucho.

Ejercicios bucle PARA (FOR)

1. Crea un programa que escriba los números del 5 al 15, ambos incluidos.
2. Crea un programa que escriba los múltiplos del 3, desde el 3 hasta el 30, usando un paso de tamaño 3.
3. Crea un programa que escriba los múltiplos del 3, desde el 3 hasta el 30, contando del uno al diez pero mostrando ese contador multiplicado por tres.
4. Crea un programa que escriba los números del 20 al 10, descendiendo.
5. Crea un programa que escriba la tabla de multiplicar del 5: desde "5 x 0 = 0" hasta "5 x 10 = 50"
6. Ejercicio de repaso propuesto 8.6: También se puede contar usando una orden "mientras" o una orden "repetir", si usas una variable como contador e incrementas (o disminuyes) su valor en cada pasada de forma manual. Compruébalo creando un programa que escriba los números del 1 al 15 usando "mientras" en vez de "para".
7. A partir del ejemplo que dibuja un rectángulo de asteriscos que aparece en el apartado "CONCLUSIONES", crea un que dibuje un cuadrado (deberá pedir sólo un dato, el lado, y ambas órdenes "para" deberán tener ese valor como límite).

8. Dibuja un triángulo creciente de asteriscos, del tamaño que indique el usuario. Por ejemplo, si escoge 4, el resultado debería ser:

CONCLUSIONES

1. El bucle **MIENTRAS** se emplea cuando las instrucciones deben **NINGUNA o varias veces**
2. El bucle **REPETIR** se emplea cuando las instrucciones deben ejecutarse como **MÍNIMO UNA VEZ**
3. El bucle **PARA** se emplea cuando **sabemos a priori cuantas veces se ejecutarán** las instrucciones
4. Las estructuras repetitivas ("bucles") se pueden incluir una dentro de otra si fuera necesario. El resultado se conoce como un **"bucle anidado"**. Por ejemplo, si se desea dibujar un rectángulo usando asteriscos, se puede plantear como escribir de forma repetitiva varias filas que, a su vez están formadas cada una por varias columnas, de forma también repetitiva, como muestra este ejemplo:

```
// Rectángulo formado por asteriscos
```

Algoritmo Rectangulo

Escribir Sin Saltar "Introduce el ancho: "

Leer ancho

Escribir Sin Saltar "Introduce el alto: "

Leer alto

Para fila<-1 **Hasta** alto **Hacer**

Para columna <- 1 **Hasta** ancho **Hacer**

Escribir Sin Saltar "*" ;

FinPara

Escribir ""; // Avance de línea tras cada fila

FinPara

FinAlgoritmo

Ejercicios de bucles:

- 1) Realizar un algoritmo que permita calcular la suma de los números ingresados mientras que el valor acumulado no supere el valor 100. Mostrar el valor acumulado antes de superar 100.
- 2) Ingresar juegos de cuatro valores cada uno. Calcular y emitir el promedio de cada juego. El proceso finaliza al encontrarse un juego cuyo primer valor es 0 (cero).
- 3) Leer una lista de números que finaliza cuando se ingresar el número 0 (cero), al finalizar emitir el valor mínimo de la lista.
- 4) Leer una lista de números que finaliza cuando se ingresar el número 0 (cero), al finalizar emitir el valor máximo de la lista.
- 5) Leer una lista de números que finaliza cuando se ingresar el número 0 (cero), al finalizar emitir el valor máximo de la lista, y la ubicación del máximo dentro de la lista. (Suponer un único máximo).
- 6) Escribir un programa que permite ingresar dos valores A y B que determinan un intervalo, luego ir acumulando los valores que se ingresan a continuación siempre y cuando estos pertenezcan al intervalo. El ingreso de números finaliza cuando ingresa el 99.
- 7) Diseñar el algoritmo para resolver una ecuación de segundo grado. El algoritmo deberá ingresar A, B y C e ir ingresando x. El programa finaliza cuando ingresa $x = 99$.
- 8) Se tienen los siguientes datos sobre nacimientos en una ciudad: sexo ("F" ó "M") y fecha de nacimiento (DD y MM). Se pide realizar un algoritmo que informe cuántos son varones y cuántas son mujeres, cuántos nacimientos hubo en el primer semestre y cuántos en el segundo. El final de lectura de datos viene dado por una lectura del sexo en blanco.
- 9) Realizar un algoritmo que determine si una serie de números ingresada por teclado es ascendente. El final de la serie viene dado por un número negativo. (Ej: 1, 5, 5, 10, 11, 12, 12, 20, -1 es una serie ascendente).
- 10) Diseñar un algoritmo que permita mostrar en pantalla una rutina de selección del siguiente menú:
 - 1.- Suma
 - 2.- Resta
 - 3.- Producto

4.- División

S.- Salir.

El usuario podrá elegir cualquier alternativa, luego ingresar A y B y realizar la operación seleccionada. Solamente con "S" podrá Salir. Tener en cuenta que si elige 4.- División deberá ingresar el denominador hasta que ingrese un valor diferente a 0 (cero). Si ingresa un número negativo o mayor que 4 deberá informar "Opción no válida".

- 11) **Un número OMIRP** es un número primo que tiene una particularidad que lo hace diferente. Si se invierten los dígitos del número, se forma otro número. Este otro número también es un número primo, por ello se los llama números OMIRP. (PRIMO al revés)

Ejemplo Se desea escribir un algoritmo que permita determinar si un número n , tiene la característica de ser un número OMIRP. Utilizando como ejemplo el número 1597:

- 1597 es número primo,
- Se invierten sus dígitos: 7951
- Se comprueba que 7951 es primo,
- Entonces el número 1597 es un número omirp.

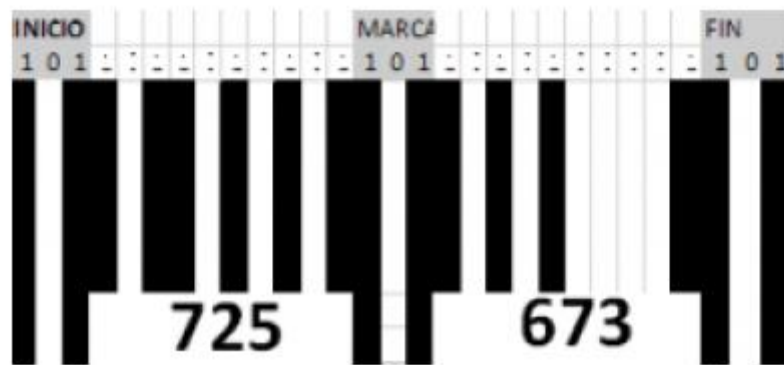
Ingresar duplas de valores formadas cada una de ellas por un carácter y un dígito. Este ingreso no debe seguir ningún orden y no debe exceder las cuatro duplas. El carácter puede tomar los siguientes valores:

- U: unidades,
- D: decenas,
- C: centenas,
- M: unidades de mil.

Calcular y mostrar el número correspondiente.

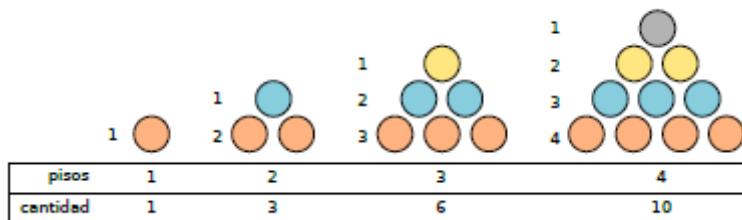
- 12) **Conversión binario a decimal** y decimal a binario: Realizar los algoritmos Binario a Decimal y Decimal a Binario usando el bucle Repetir-MientrasQue.
- 13) Crear un algoritmo para convertir un número **de base numérica genérica** x a base y . Sugerencia, convierta el número de la base X primero a decimal y luego transforme el resultado a la siguiente base numérica Y .
- 14) **Código de barras:** El código de barras utiliza líneas paralelas verticales (barras y espacios) que representan información en su equivalente binario. El código es muy usado en los puntos de ventas y es "leído" por un dispositivo láser (scanner). Para facilitar la lectura por scanner se usa el método de "simbología discreta", en el que se marca el inicio, separación y fin de los datos con la secuencia barra/espacio/ barra (101) por cada grupo de 10 bits (dígitos binarios). Elabore un algoritmo que permita cambiar un código de producto

conformado por dos números de 3 cifras a su equivalente en código de barras usando simbología discreta. Referencia: http://es.wikipedia.org/wiki/C%C3%B3digo_de_barras



15) Números triangulares:

Un faraón le pide a su ingeniero Real que calcule los materiales necesarios para construir su pirámide. El ingeniero real propone escribir un algoritmo que le permita encontrar el término n-ésimo de la secuencia de números triangulares. Un número triangular puede entenderse como el número de elementos usados para formar una pirámide plana, como se muestra en la figura:



Desarrollo:

Usando la analogía de la construcción de una pirámide, se observa que para construir un piso se usa un bloque, para añadir el piso 2 se usan 2 bloques, para añadir el piso 3 se usan 3 bloques, etc.

Al principio se le pregunta al Faraón cuantos pisos desea construir. El ingeniero real ingresa esa información en el programa, que se almacena en la variable llamada CUÁNTOS. Para iniciar los trabajos, se anuncia la construcción del primer piso y que se han usado 0 bloques. En el procedimiento de construcción de la pirámide, PISO es una variable tipo contador; USADOS es una variable tipo acumulador que registra los bloques usados en la construcción de cada piso.

Se construye un nuevo piso, acumulando los bloques usados hasta que se hayan construido los pisos de la variable cuántos. La variable de salida que muestra el valor resultante es: USADOS.

16) Averiguar números múltiplos de 9

Un entero es divisible para 9 si lo es la suma de sus cifras. Conociendo ésta propiedad, realice un algoritmo para determinar si un número n es divisible para 9.

Por ejemplo:

- Numero= 1492,-> $1+4+9+2=16$ -> $1+6=7$;
 - Resultado=como 7 no es divisible por 9, 1492 no es divisible por 9.
- Numero=1548-> $1+5+4+8=18$ -> $1+8=9$
 - Resultado= como 9 es divisible por 9, 1548 es divisible por 9.

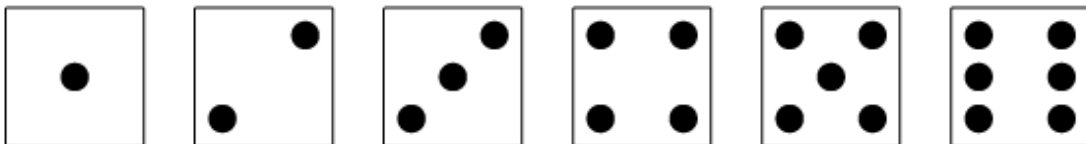
Bucles con números Aleatorios

Números aleatorios

Para simular un número de azar, por ejemplo el obtenido al lanzar un dado, se recurre al concepto de números aleatorios. Un número aleatorio se define como un número real cualquiera en el rango $[0,1)$



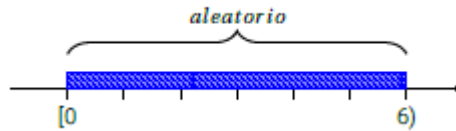
Para utilizar el número aleatorio se debe convertir al rango apropiado del número a simular, por ejemplo, para simular un dado se escribiría en el algoritmo:



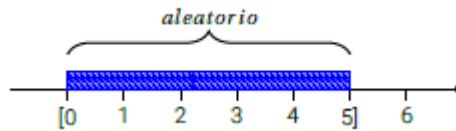
$$dado \leftarrow \text{entero}(\text{aleatorio} * 6) + 1$$

Como el dado tiene 6 caras se multiplica por 6, obteniendo un real de $[0,6)$. No se incluye el

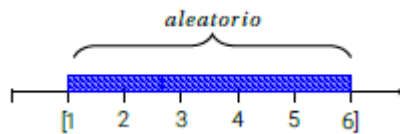
6



Se extrae solo la parte entera para obtener un número entero [0,5]



Para que el resultado sea [1,6], se le suma 1



Juego Tirar Dados

Realizar un programa que permita generar tiradas de un dado. Para ello es posible utilizar la función `azar(N)`. El argumento `N` de la función sirve para indicar que genere un número aleatorio entero con un rango entre 0 y `N-1`.

Ejemplo de salida:

```
PSeInt - Ejecutando proceso TIRADAS_DADO
*** Ejecución Iniciada. ***
Ingrese la cantidad de tiradas a calcular:> 6
Tiradas:
4,6,4,4,6,5,5,*** Ejecución Finalizada. ***
```

Juego de Generala

Modificar el programa anterior de forma que calcule el resultado de una tirada de la generala.

La generala se juega con 5 dados. Considerar los siguientes casos:

- Si todos los números generados son iguales, es GENERALA.
- Si 3 números son iguales y 2 son iguales, es FULL.
- Si 4 números son iguales es POKER.
- Si todos los números son crecientes, es ESCALERA

Juego: Precio del Petróleo

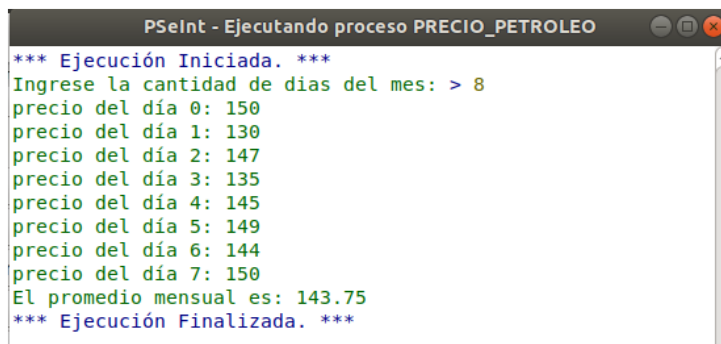
Realice un algoritmo para simular el precio del barril de petróleo durante un mes de 30 días, suponiendo que son valores enteros que fluctúan en forma aleatoria entre \$ 130 y \$ 150 y se obtenga las siguientes respuestas:

- a) El promedio del precio del petróleo.
- b) ¿Cuál fue el día en el que estuvo más barato el barril de petróleo?

Desarrollo:

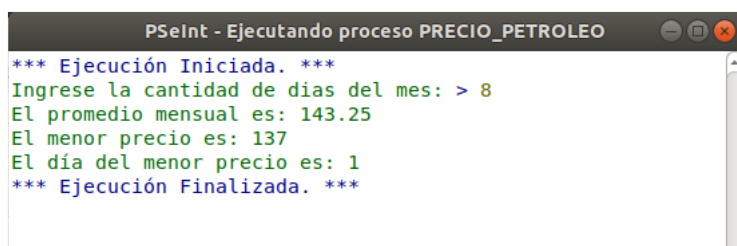
Para iniciar el algoritmo, se puede considerar como variable de entrada los días del mes, o asignarles directamente 30 días. La primera aproximación al problema para responder el literal a) consiste en generar números aleatorios en el rango [130, 150] y acumular sus valores para el promedio. Será necesario disponer de un contador para controlar el número de veces que se generan los precios de forma aleatoria en el bucle de repetición. Una de las formas de resolver el problema es con un lazo REPETIR MIENTRAS.

A continuación se muestra un ejemplo de salida del programa



```
PSeInt - Ejecutando proceso PRECIO_PETROLEO
*** Ejecución Iniciada. ***
Ingrese la cantidad de días del mes: > 8
precio del día 0: 150
precio del día 1: 130
precio del día 2: 147
precio del día 3: 135
precio del día 4: 145
precio del día 5: 149
precio del día 6: 144
precio del día 7: 150
El promedio mensual es: 143.75
*** Ejecución Finalizada. ***
```

Para la pregunta b) es necesario analizar la manera de encontrar el día con el precio más barato. En este caso se utilizará un algoritmo para búsqueda del número menor, que consiste en iniciar con el supuesto para el valor menor de precio y día, comparando contra el precio de cada día y de ser necesario se cambian los valores menores.



```
PSeInt - Ejecutando proceso PRECIO_PETROLEO
*** Ejecución Iniciada. ***
Ingrese la cantidad de días del mes: > 8
El promedio mensual es: 143.25
El menor precio es: 137
El día del menor precio es: 1
*** Ejecución Finalizada. ***
```

Tiro de penalties

El juego que se plantea consiste en 5 lanzamientos por parte de los jugadores que golpean el balón, los cuales pueden decidir lanzar en cualquiera de las seis secciones del arco (1: arriba a la derecha, 2: arriba al centro, 3: arriba a la izquierda, 4: abajo a la izquierda, 5: abajo al centro, 6: abajo a la derecha). En cada lanzamiento, el portero decide donde ubicarse para atajar el tiro y no tiene oportunidad de cubrir otra sección, si éste coincide con la ubicación donde disparó el jugador, entonces el lanzamiento fue atajado o fallado, caso contrario se marcó un GOL.

Escriba un algoritmo que simule un juego de 5 lanzamientos de penales, en donde la sección del arco donde cada jugador lanza es decidido por el usuario y la sección cubierta por el arquero es simulado por el computador (aleatoria).

Al final presente la siguiente información:

- Cantidad de goles conseguidos.
- Cantidad de penalties fallados.
- La cantidad de goles realizados en la parte derecha, central e izquierda del arco.
 - La ubicación del arco (derecha, centro o izquierda) por donde ingresaron más goles. Suponga que existe una sola.
 - La ubicación del arco (derecha, centro o izquierda) por donde no ingresaron goles. Suponga que existe una sola.

