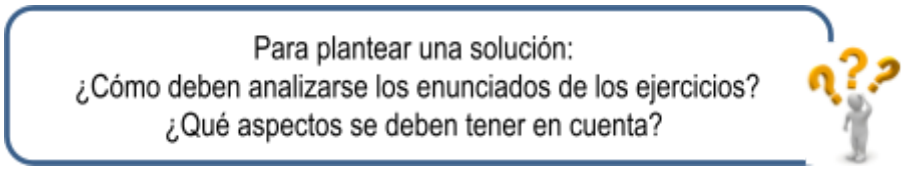
Tema: Estructuración de Datos – PSeInt

Objetivo:

- Resolver problemas diseñando algoritmos que involucren el uso de estructuras de datos. La implementación se realizará respetando la sintaxis de PSeInt.

Aspectos a tener en cuenta para pensar soluciones que incorporen la utilización de variables estructuradas (arreglos).



1- Las siguientes preguntas pueden ayudarte a encontrar una solución para cada ejercicio:

   - ¿Cuáles son los **procesos** (tareas) **que debo implementar** para dar solución a lo planteado en el enunciado? ¿Qué debe hacer el **algoritmo**?
   - ¿Qué **datos** iniciales se le piden al usuario? ¿Debo realizar **controles** sobre los mismos?
   - ¿Qué datos deben ser resultado del proceso? O de los procesos.
   - ¿Necesito utilizar una **variable estructurada** o más de una? ¿Qué dice el enunciado al respecto?
   - ¿De qué **dimensiones** y de qué **tipo** deben ser las variables estructuradas a utilizar?
   - ¿Qué **controles** debo realizar al **manipular las estructuras de datos**?

2- También es útil graficar la/s estructura/s y analizar los diferentes procesos que se realizan sobre la/s misma/s, analizando:

   - **Cantidad de índices** necesarios para trabajar. ¿puedo usar el mismo índice para todas las estructuras? ¿Debo usar más de un índice?
   - **Condiciones a evaluar y controles** que debo tener en cuenta para manejar cada arreglo. Este aspecto puede depender de la dimensión de cada estructura y del espacio real ocupado por datos válidos, restricciones en el enunciado, etc.

3- En general los diferentes procesos a realizar en una estructura de datos, en nuestro caso en un arreglo se implementan por medio de iteraciones, para lo cual es importante analizar:

- ¿Cuántas veces se debe repetir la secuencia de pasos establecidas para el proceso?
- ¿Utilizaremos la sentencia MIENTRAS o la sentencia PARA? ¿De qué depende?
- ¿Cuál es la condición o las condiciones que deben evaluarse en la iteración? ¿Qué variables y qué datos utilizo para implementarlas?
- ¿Es necesario forzar la salida de la iteración en algún momento en particular? ¿Cómo es posible hacerlo?

Enunciados

1. Analizar el siguiente problema:

Problema: Diseñar un algoritmo que permita ingresar los datos necesarios para calcular, el monto de antigüedad, a pagar a los empleados de la fábrica Nexo.

Suponga que en dicha empresa el cálculo se realiza considerando el monto básico de sueldo y el % a pagar por categoría para antigüedad mayor a 10 años:

%	Categoría
1.05	Planta
1.35	Administración

Responder:

a. ¿Cuáles son los datos que se deben almacenar? ¿De qué tipo son? ¿Especificar un nombre para el conjunto de datos dados.

b. Decidir la cantidad de estructuras a utilizar para almacenar los datos, el tipo y la dimensión de cada una.

c. Realizar las declaraciones correctas.

a) Los datos que debemos almacenar, por cada empleado serán: *año de ingreso* (para calcular la antigüedad) de tipo entero y *categoría* de tipo caracter. El nombre para el conjunto de datos sería *Antiguedad de Empleados*.

b) Decido trabajar con **una única estructura** de tipo **entero**, la dimensión será de 100 considerando un total de empleados de 50 y guardando 2 datos por empleados, ***año de ingreso entero*** y ***categoría*** (codificando 1 para Planta y 2 para Administración)

c) Definir **AntEmpl** Como Entero

Dimensión **AntEmpl[100]**

2003	1	2010	2						
1	2	3	4	5	…	97	98	99	100

Nota: Los datos del empleado 1 serán los datos de la posición 1 y 2, los del empleado 2 serán los de la posición 3 y 4. Así mismo los datos del empleado 50 estarán en las posiciones 99 y 100.

2. Diseñar un algoritmo que permita el ingreso de hasta 10 notas, luego calcular el promedio y mostrar el resultado. El algoritmo debe consultarle al usuario si desea continuar ingresando.

Versión 1

t1: definir variables para almacenar hasta 10 notas y para calcular el promedio.

t2: ingresar las notas que el usuario desee controlando que no supera las 10.

t3: calcular el promedio de las notas ingresadas.

t4: mostrar el promedio calculado.

Versión 2

t1.1: definir un arreglo de tipo real con dimensión 10

t1.2: definir una variable de tipo entero para utilizar como índice del arreglo

t1.3: definir una variable de tipo real para calcular el promedio

Mientras el usuario desee ingresar notas y no supere las 10 notas

t2.1.1: pedirle al usuario que ingrese 1 nota

t2.1.2: guardar la nota ingresada en la posición del arreglo que indique el índice

t2.1.3: controlar que la nota sea válida

t2.1.4:actualizar índice

t3.1: recorrer el arreglo sumando las notas ingresadas

t3.2: dividir el total sumado por el total de notas ingresadas

t4.1: mostrar el cartel El promedio es:

t4.2: mostrar el promedio calculado

versión Final

```
Algoritmo promedio
//T1: Definir variables
definir notas como real
dimension notas[10]
definir sumaNotas,prom como Real
definir i,j, seguir como entero
i←1
seguir←1
//T2: ingresar datos (HASTA 10 notas y mientras que el usuario quiera seguir ir
mientras i≤10 y seguir == 1 hacer
    Escribir "ingrese una nota"
    leer notas[i]
    //control de nota válida
    Mientras notas[i]<0 o notas[i]> 10 hacer
        Escribir "ingrese una nota en el rango 0 - 10"
        leer notas[i]
    FinMientras
    //consulta al usuario
    escribir "desea continuar ingresando notas?, Ingrese 1= Si ó 2=No"
    leer seguir
    Mientras seguir ≠ 1 y seguir ≠ 2 Hacer
        escribir "desea continuar ingresando notas?, Ingrese 1= Si ó 2=No"
        leer seguir
    FinMientras
    //conteo de nota ingresada
    i←i+1
finMientras
i←i-1
sumaNotas←0
//T3: Calcular promedio
Para j ← 1 hasta i con paso 1 hacer
    sumaNotas← sumaNotas + notas[j]
FinPara
prom ← sumaNotas / i
//T4: mostrar resultados
Escribir "el promedio de notas es: ", prom
FinAlgoritmo
```

3. Según el siguiente problema:

Se necesita almacenar la información de hasta 100 empleados de la empresa "Gráfica SRL". De cada empleado se registra: *nombre y apellido, año de nacimiento, sexo, antigüedad y categoría (planta, administración, gerencias).* Se pide diseñar y dibujar de qué manera considera la estructuración de los datos.

Nota: NO se pide diseñar el algoritmo. Sólo hay que decidir y especificar claramente cómo almacenar los datos.

1- Del análisis del enunciado decimos que:

- No se menciona ninguna restricción con respecto a la cantidad de arreglos que debemos utilizar.
- La **entidad** con la que trabajaremos es **Empleado**. (Una entidad es la representación de un objeto o concepto del mundo real. Ejemplos de nombres de entidades: Alumno, Empleado, Artículo, Noticia, etc. Diremos que una entidad se representa por una tupla de datos (un conjunto de datos)

- De cada empleado se registra: nombre y apellido, año de nacimiento, sexo, antigüedad y categoría (planta, administración, gerencias).
- En este caso se especifica la cantidad total de empleados, 100 empleados.

2- Del análisis de los datos decimos que los datos a almacenar son:

- nombre y apellido (tipo caracter)
- año de nacimiento (tipo entero)
- sexo (tipo¿?)
- antigüedad (tipo entero)
- categoría (planta, administración, gerencias). (tipo ¿?)

Los datos sexo y categoría podrían codificarse como enteros:

- sexo: 1- Femenino, 2: Masculino, 3- Otro
- categoría: 1: Planta, 2: Administración, 3: Gerencias

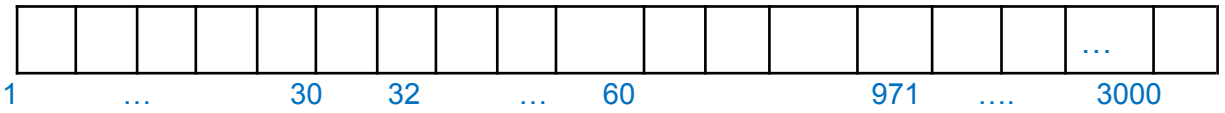
3- Estructuración de los Datos:

Se propone utilizar 2 arreglos,uno para almacenar los nombres de los empleados y otro para almacenar el resto de datos (año de nacimiento, sexo, antigüedad, categoría).

4- Diseño:

Arreglo: Nbres_empleados:

- Supongo que **cada 30 posiciones** se almacenará el nombre de un empleado
- Dimensión del arreglo: 3000 (30 x 100)
- Tipo del arreglo: caracter (porque los datos que guardaré son letras, valores incluidos en el tipo caracter)



Arreglo: Otros_datos_empleados:

- Dimensión del arreglo: 400 (4 x 100)
- Tipo del arreglo: entero (porque los datos *año de nacimiento, sexo, antigüedad, categoría* que voy a guardar son numéricos, donde sexo: 1- Femenino, 2: Masculino, 3- Otro . Categoría: 1: Planta, 2: Administración, 3: Gerencias

Añonac_E	Sexo_E	Anti_E	Categ_E	.	Añonac_E	Sexo_E	Anti_E	Categ_E
1	2	3	4		397	398	399	400

Ejemplo:

Empleado 1:

- Carla Sosa
- 1978
- Femenino
- 25
- Gerencias

Arreglo: Nbres_empleados:

‘c’	‘a’	‘r’	‘l’	‘a’	‘ ‘	‘s’	‘o’	‘s’	‘a’	‘#’		...				…	
1	2	3…	4	5	6	7	8	9	10	…. **30**		…	60				3000

Arreglo: Otros_datos_empleados:

1978	1	25	3	…….						...				…	
1	2	3…	**4**	5	30	….	60								400

4. Diseñar un algoritmo que permita ingresar no más de 30 **números enteros positivos** (la cantidad de números la decide el usuario). Los números pares se almacenarán en posiciones pares y los números impares en posiciones impares. Posteriormente, se debe mostrar por pantalla una secuencia de números respetando el siguiente criterio:

   - mostrar los números impares en orden inverso al que fueron ingresados.

Versión 1

T1: Pedir el ingreso de no más de 30 **números enteros positivos**, controlando que los números pares se almacenen en posiciones pares y los números impares en posiciones impares de la estructura a utilizar.

T2: Mostrar los números impares en orden inverso al que fueron ingresados.

Versión 2

T1.1: Definir un arreglo de tipo entero y dimensión 30 para almacenar los nros enteros positivos ingresados por el usuario.

T1.2: Definir la variable entera **cantidad** para almacenar la cantidad de números a ingresar

T1.3: Definir la variable entera **cont** para contador de números y las variables enteras posPar y posImpar para índices en el arreglo de posiciones pares e impares según corresponda

T 1.4: Inicializar **cont, posPar,posImpar**

T1.5: Pedir al usuario que ingrese en **cantidad** cuantos numeros va a ingresar, **controlando** que no sean más de 30.

Mientras **cont** sea menor o igual a **cantidad**

T1.4.1: Pedir al usuario que ingrese un número, **controlando** que sea positivo

Si el número es par

T1.4.2.1: Almacenar el número en la posición **posPar** del arreglo

T1.4.2.2: Actualizar **posPar**

Sino

T1.4.3.1:Almacenar el número en la posición **posImpar** del arreglo

T1.4.3.2: Actualizar **posImpar**

T1.4.4: Incrementar cont en 1

Mientras **posImpar** sea mayor o igual a 1

T2.1.1: Mostrar el arreglo en la posición **posImpar**

T2.1.2: Actualizar **posImpar**

Versión Final

```
Algoritmo  EJERCICIO3
   Definir arr, cantidad,Cont, num, posPar, posImpar Como Entero
   //el arreglo debe ser de 60 posiciones para contemplar el caso de que el usuario ingrese 30pares o 30 impares
   Dimension arr[60]
   //inicializaciones de indices
   posImpar ← 1
   posPar ← 2
   //contador
   Cont ←1
   //SE LE PIDE LA CANTIDAD A INGRESAR AL USUARIO//
   Escribir "¿CUANTOS NÚMEROS VA A INGRESAR?"
   Leer cantidad
   //CONTROL PARA QUE NO INGRESE MÁS DE 30 NÚMEROS //
   Mientras (cantidad >30) Hacer
      Escribir "INGRESE UNA CANTIDAD MENOR O IGUAL A 30"
      Leer cantidad
   FinMientras
   // SE PIDE QUE INGRESE LO NÚMEROS ITERANDO HASTA CANTIDAD//
   Mientras Cont ≤ cantidad Hacer
      Escribir "INGRESE ", Cont," NUMERO"
      Leer num
      Mientras (num≤ 0) Hacer
         Escribir "Ingrese nuevamente un número positivo"
         Leer num
      FinMientras
      //SI EL NÚMERO ES PAR SE INGRESA EN UNA POSICIÓN PAR//
      Si (num MOD 2 = 0) Entonces
         arr[posPar] ← num
         posPar ← posPar + 2
         // SI EL NÚMERO ES IMPAR SE LO INGRESA EN UNA POSICIÓN IMPAR//
      SiNo
         arr[posImpar] ← num
         posImpar ← posImpar + 2
      FinSi
      Cont ← Cont+1
   FinMientras
   //ETAPA DE MUESTRA
   Si posImpar == 1 entonces
      Escribir "No se ingresaron impares"
   sino
      //Se LE RESTA A POSIMPAR PARA QUEDAR EN LA ÚLTIMA POSICIÓN OCUPADA EN EL ARREGLO //
      posImpar ← posImpar - 2
      //MUESTRA LOS VALORES INGRESADOS DE FORMA INVERSA AL INGRESADO
      Mientras posImpar ≥ 1 Hacer
         Escribir "VALOR INGRESADO IMPAR: ", arr[posImpar]
         posImpar ← posImpar - 2
      FinMientras
   FinSi
   FinAlgoritmo
```