Teoría

Pseudocódigo: Subalgoritmos

Resolución de Problemas y Algoritmos
Fundamentos de la programación

Ingeniería en Computación (TU y TFA)
Ingeniería en Minas (TU)
Profesorado en Ciencias de la Computación (TU y TFA)

Teoría

Pseudocódigo: Subalgoritmos

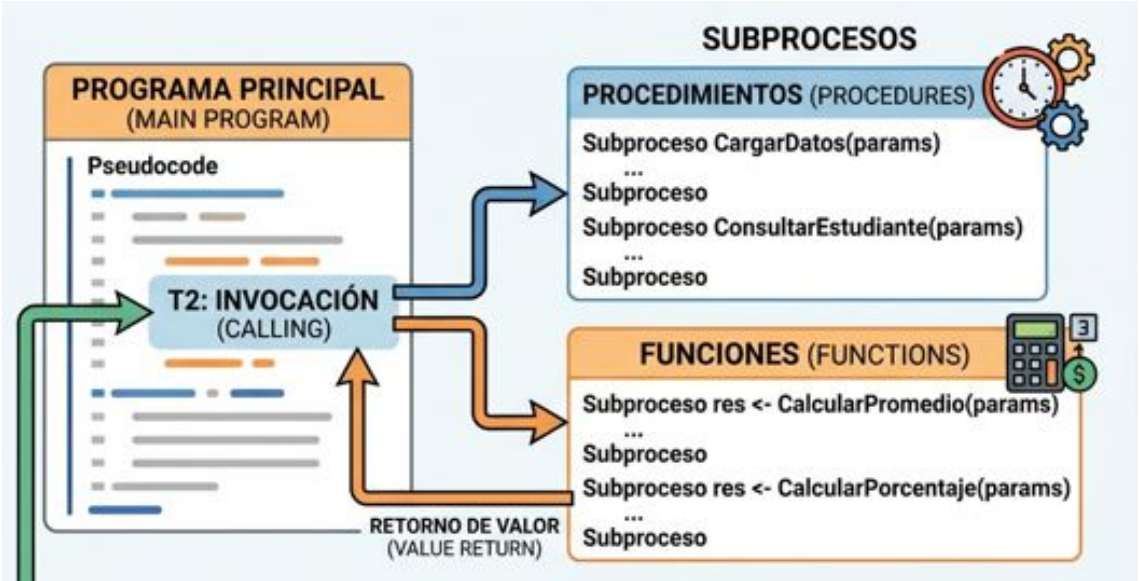
- ✔ **Modularización.**
- ✔ **Diferencia entre Funciones y Procedimientos.**
- ✔ **Sintaxis en PSeInt de Funciones y Procedimientos.**
- ✔ **Definición e Invocación de Funciones y Procedimientos.**
- ✔ **Ejecución con subalgoritmos.**
- ✔ **Tipos de parámetros: por valor y por referencia.**
- ✔ **Arreglos como parámetros.**
- ✔ **Ámbitos de variables: locales o globales.**

Modularización.

Es una técnica de programación que permite dividir un problema en varios pequeños problemas más simples para llegar a la solución.

Podríamos decir que la programación modular es un conjunto de **subprogramas o subalgoritmos** que se comunican entre sí para dar una solución simplificada.

La comunicación entre **subalgoritmos** se realiza por medio de **parámetros** (variable del subprograma).



Modularización.

Intentar resolver un problema en un solo bloque de código genera confusión, errores y dificulta la lectura.

Modularización

Dividir el problema en subprogramas (subalgoritmos) más simples que se comunican entre sí. La complejidad se transforma en un sistema de piezas manejables.

Modularización.

Legibilidad

Aumenta drásticamente la comprensión del programa y facilita encontrar errores (depuración).

Reutilización

Evita escribir lo mismo varias veces. Ej: Calcular el promedio de edad para estudiantes de IC, IM y PC.

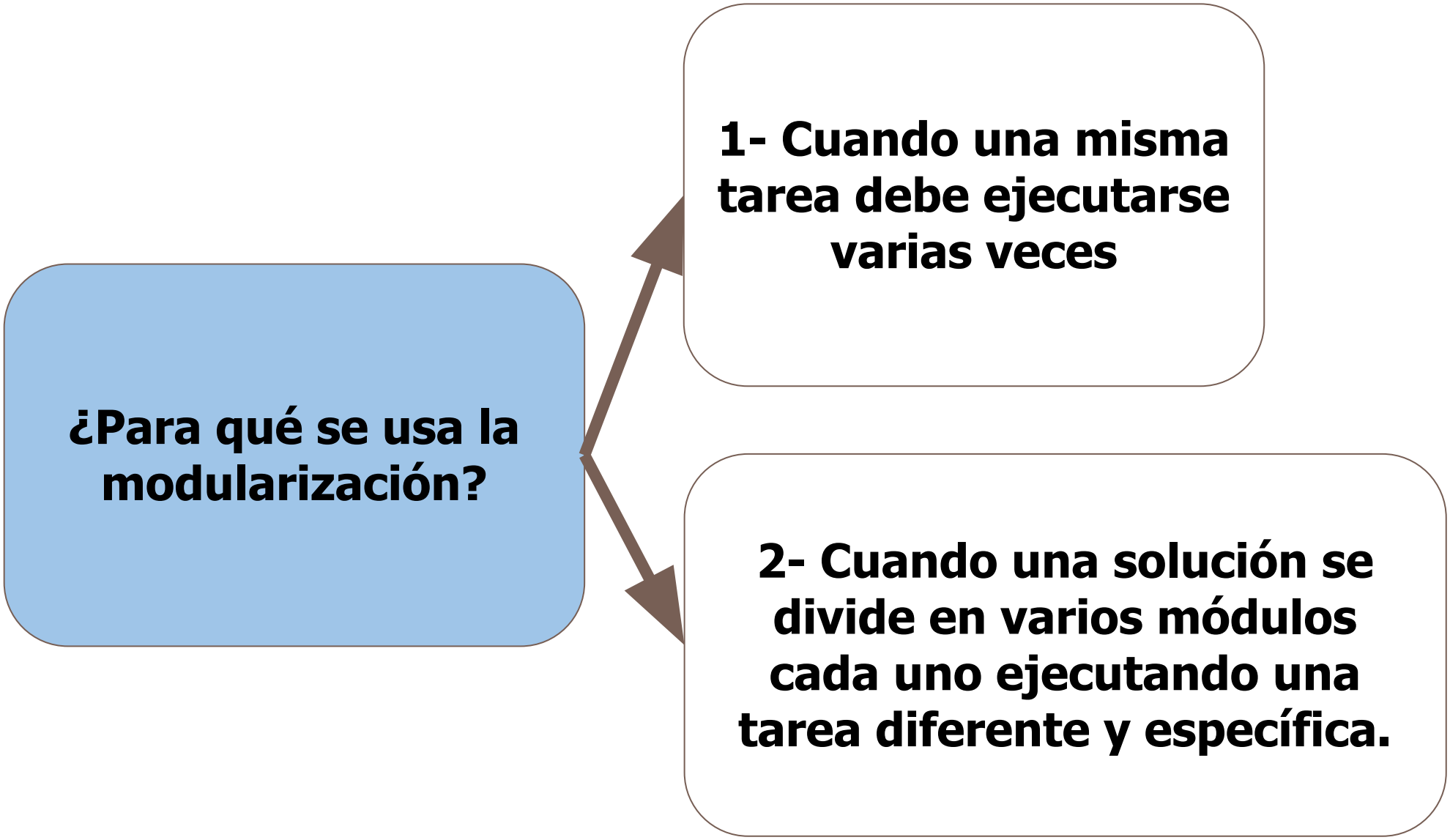
Trabajo en Equipo

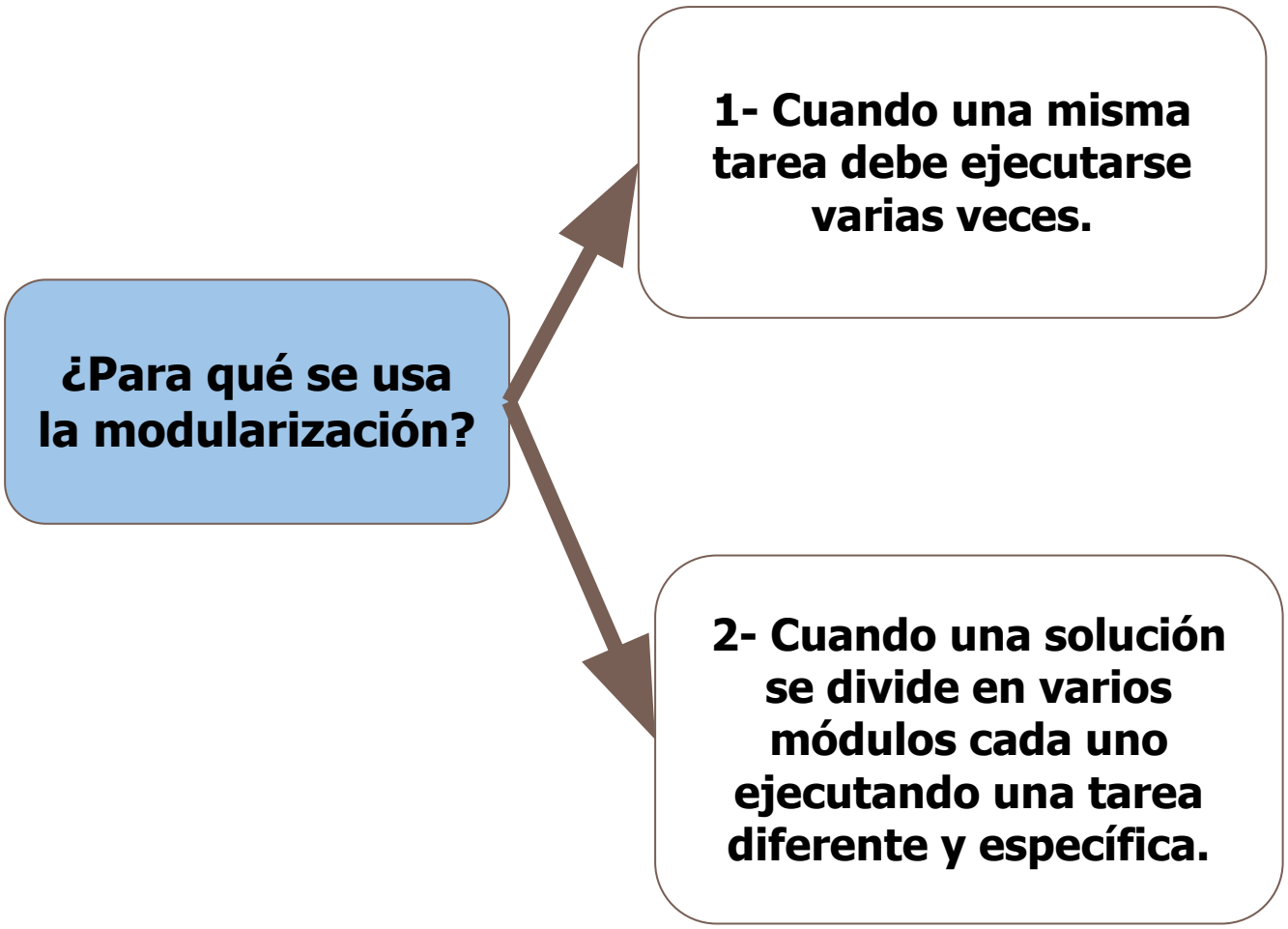
Permite que varios programadores desarrollen diferentes piezas del código al mismo tiempo.

Especialización

Cada módulo ejecuta una tarea específica. Ej:

1. Ingresar datos de 150 empleados.
2. Calcular promedios.
3. Mostrar resultados.


¿Para qué se usa la modularización?
1- Cuando una misma tarea debe ejecutarse varias veces
2- Cuando una solución se divide en varios módulos cada uno ejecutando una tarea diferente y específica.



Por ejemplo:

- **Calcular el promedio de edad de los estudiantes de la carrera de IC.**
- **Calcular el promedio de edad de los estudiantes de la carrera de IM.**
- **Calcular el promedio de edad de los estudiantes de la carrera de PC.**

Por ejemplo:

- **Ingresar los datos personales de 150 empleados.**
- **Calcular el promedio de edad de los empleados y cuántos son mayores de 50 años.**
- **Mostrar los datos completos de todos los empleados.**

Diferencia entre Procedimiento y Función.

En Lenguaje de Diseño el concepto de **Modularización** se implementa a través de **SUBALGORITMOS**, con **Funciones** y **Procedimientos.**

FUNCIONES

Toda **función** es un subalgoritmo que TIENE que retornar UN resultado y opcionalmente se le puede indicar un conjunto de datos (**parámetros**) para que pueda funcionar.

Solamente retorna UN único valor.

Se utiliza para realizar cálculos y retorna UN resultado.

PROCEDIMIENTOS

Un procedimiento es un subalgoritmo que NO retorna un resultado y opcionalmente se le puede indicar un conjunto de datos (**parámetros**) para que pueda funcionar.

Funciones

Diseñadas para calcular. **TIENEN** que retornar **UN** único resultado. (Ej: Calcular el porcentaje de días no laborables).

Procedimientos

Diseñados para ejecutar acciones. **NO** retornan un resultado directo. (Ej: Mostrar la lista de habitaciones libres).

Actividad 1: Analizar el enunciado de cada módulo y decidir si su solución debe implementarse como **Función (F)** o **procedimiento (P)**.

Por ejemplo:

1. P o F?
2. …

1. Ingresar el dato de edad y altura de un nuevo estudiante.
2. Calcular el porcentaje de días no laborables de un mes.
3. Mostrar la lista de habitaciones libres de un hotel.
4. Determinar si un número es par, retornando un VERDADERO si es o un FALSO en caso contrario.
5. Ingresar 20 números enteros positivos.
6. Mostrar los caracteres ingresados en un arreglo llamado NOM.
7. Reemplazar todo caracter ‘a’ por un ‘#’.
8. Contar cuántos caracteres se reemplazaron.

Sintaxis en PSeInt de Funciones y Procedimientos.

FUNCIONES

SubAlgoritmo <variable_de_retorno> ← <Nombre> (Parámetro1, Parámetro2, ..)

Definir <variable_de_retorno> como <tipo de dato>
<Declaración de variables locales>
<Sentencias>

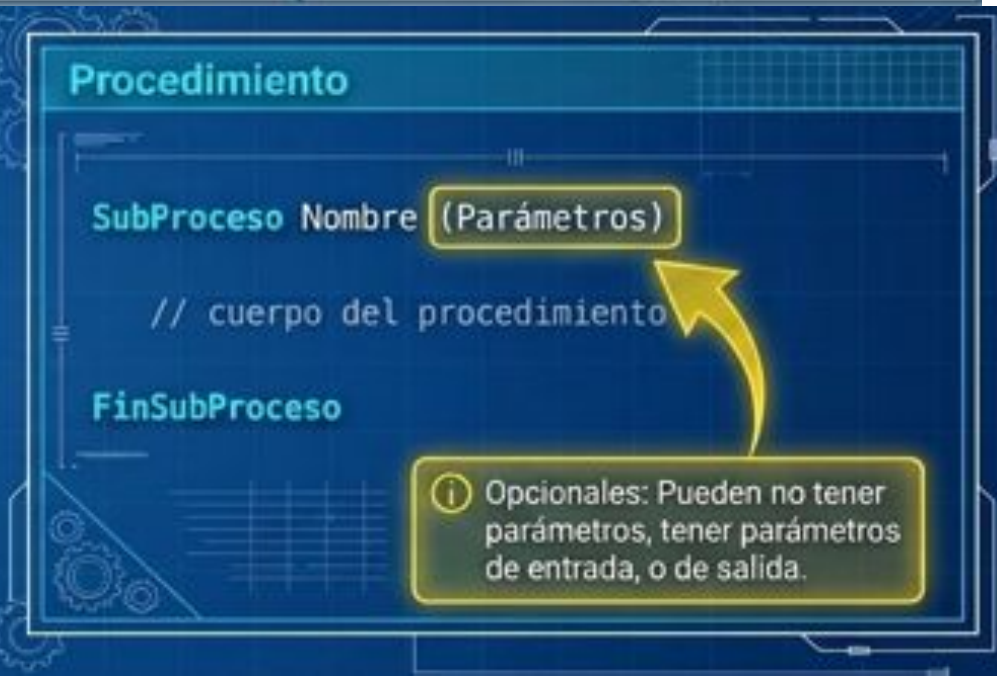
// NO OLVIDAR asignar un valor <variable_de_retorno>

FinSubalgoritmo

PROCEDIMIENTOS

SubAlgoritmo <Nombre> (Parámetros1, Parámetro2, …)
<Declaración de variables locales>
<Sentencias>
FinSubalgoritmo

Característica	Funciones	Procedimientos
Retorna un valor al programa	✅ Obligatorio (Solo UN valor)	❌ No retorna valor (directamente)
Uso Principal	Cálculos matemáticos o lógicos (Ej: Determinar si es múltiplo de 5).	Ejecución de tareas o acciones (Ej: Ingresar 20 números, Mostrar un arreglo).



FUNCIONES

```
SubProceso Pr <-Promedio(n1 , n2)
    Definir Pr Como Real
    Pr<-(n1 + n2)/2.0
FinSubProceso
```

PROCEDIMIENTOS

```
SubProceso MostrarPromedio(Pr)

    escribir "El promedio calculado es: ", Pr
FinSubProceso
```

Actividad 2: Completar la tabla con los valores correspondientes

Nombre de la función	
Nombre del procedimiento	
Parámetro/s de función	
Parámetro/s del procedimiento	
variable de retorno	

Definición e invocación de subalgoritmos.

Todos los **subalgoritmos o subprocesos** deben definirse antes del **algoritmo principal** y antes de ser invocados.

¿Cuántos subalgoritmos hay definidos? ¿Cómo los identifico?

```
1  SubProceso ingreso(nro Por Referencia)
2      Escribir "Ingrese un nro"
3      Leer nro
4  FinSubProceso
5
6  SubProceso aux <- calculo(nro)
7      Definir aux Como Entero
8      Si nro < 0 Entonces
9          aux <- abs(nro)
10     Sino
11         aux <- nro ^ 2
12     FinSi
13 FinSubProceso
14
15 Proceso p5_ej3
16     Definir nro, val, k Como Entero
17     Dimension val[3]
18
19     Para k <- 1 Hasta 3 Con Paso 1 Hacer
20         ingreso(nro)
21         val[k] <- calculo(nro)
22         Escribir "El nro ingresado fue: ", nro
23         Escribir "El número calculado fue: ", val[k]
24     FinPara
25 FinProceso
```

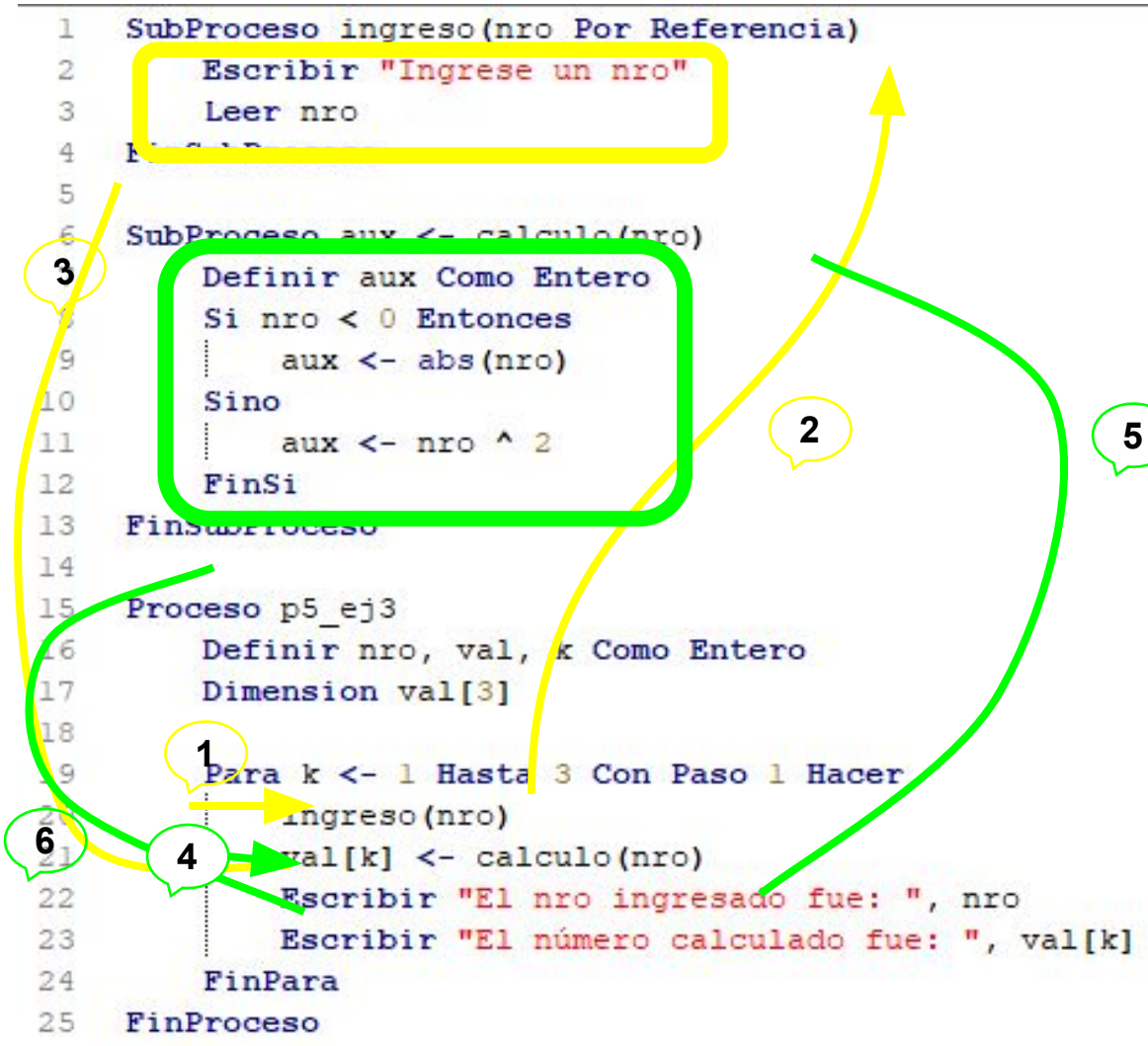
Para que las acciones descriptas en el subalgoritmo sean ejecutadas, es necesario que el mismo sea **invocado** desde el algoritmo principal o desde otro subalgoritmo.

```
1  SubProceso ingreso(nro Por Referencia)
2      Escribir "Ingrese un nro"
3      Leer nro
4  FinSubProceso
5
6  SubProceso aux <- calculo(nro)
7      Definir aux Como Entero
8      Si nro < 0 Entonces
9          aux <- abs(nro)
10     Sino
11         aux <- nro ^ 2
12     FinSi
13 FinSubProceso
14
15 Proceso p5_ej3
16     Definir nro, val, k Como Entero
17     Dimension val[3]
18
19     Para k <- 1 Hasta 3 Con Paso 1 Hacer
20         ingreso(nro)
21         val[k] <- calculo(nro
22         Escribir "El nro ingresado fue: "
23         Escribir "El número calculado fue: ", val[k]
24     FinPara
25 FinProceso
```

Invocación

Invocación

¿Cómo es la ejecución con subalgoritmos?




```
1   SubProceso ingreso(nro Por Referencia)
2       Escribir "Ingrese un nro"
3       Leer nro
4   FinSubProceso
5
6   SubProceso aux <- calculo(nro)
7       Definir aux Como Entero
8       Si nro < 0 Entonces
9           aux <- abs(nro)
10      Sino
11          aux <- nro ^ 2
12      FinSi
13  FinSubProceso
14
15  Proceso p5_ej3
16      Definir nro, val, k Como Entero
17      Dimension val[3]
18
19      Para k <- 1 Hasta 3 Con Paso 1 Hacer
20          ingreso(nro)
21          val[k] <- calculo(nro)
22          Escribir "El nro ingresado fue: ", nro
23          Escribir "El número calculado fue: ", val[k]
24      FinPara
25  FinProceso
```

1. Se invoca al procedimiento **ingreso**.

2. Se ejecutan las acciones del cuerpo del subalgoritmo **ingreso**

3. La ejecución retorna a la sentencia siguiente a la invocación de **ingreso.**

4. Se invoca a la función **calculo**.

5. Se ejecutan las acciones del cuerpo del subalgoritmo **calculo**.

6. La ejecución retorna a la sentencia siguiente a la invocación de **calculo.**


Programa principal
1. Invocación
El programa principal llama al subalgoritmo por su nombre y pausa su ejecución.
procesoA
2. Ejecución
Se ejecutan las acciones dentro del cuerpo del subalgoritmo.
3. Retorno
La ejecución regresa al programa principal, exactamente a la sentencia siguiente a la invocación.

```
SubProceso ingreso(nro Por Referencia)
    Escribir "Ingrese un nro"
    Leer nro
FinSubProceso

SubProceso aux <- calculo(nro)
    Definir aux Como Entero
    Si nro < 0 Entonces
        aux <- abs(nro)
    Sino
        aux <- nro ^ 2
    FinSi
FinSubProceso

Proceso p5_ej3
    Definir nro, val, k Como Entero
    Dimension val[3]

    Para k <- 1 Hasta 3 Con Paso 1 Hacer
        ingreso(nro)
        val[k] <- calculo(nro)
        Escribir "El nro ingresado fue: ", nro
        Escribir "El número calculado fue: ", val[k]
    FinPara
FinProceso
```

Procedimiento

Función

Programa principal

Invocación a Procedimiento

Invocación a Función

Se invoca al subalgoritmo a través de su **nombre**, seguido de la **lista de parámetros actuales**, los cuales deben coincidir en cantidad y orden con los parámetros formales.

```
1   SubProceso ingreso(nro Por Referencia)
2       Escribir "Ingrese un nro"
3       Leer nro
4   FinSubProceso
5
6   SubProceso aux <- calculo(nro)
7       Definir aux Como Entero
8       Si nro < 0 Entonces
9           aux <- abs(nro)
10      Sino
11          aux <- nro ^ 2
12      FinSi
13  FinSubProceso
14
15  Proceso p5_ej3
16      Definir nro, val, k Como Entero
17      Dimension val[3]
18
19      Para k <- 1 Hasta 3 Con Paso 1 Hacer
20          ingreso(nro)
21          val[k] <- calculo(nro)
22          Escribir "El nro ingresado fue: ", nro
23          Escribir "El número calculado fue: ", val[k]
24      FinPara
25  FinProceso
```

Invocación

```
1   SubProceso ingreso(nro Por Referencia)
2       Escribir "Ingrese un nro"
3       Leer nro
4   FinSubProceso
5
6   Funcion aux <- calculo(nro)
7       Definir aux Como Entero
8       Si nro < 0 Entonces
9           aux <- abs(nro)
10      Sino
11          aux <- nro ^ 2
12      FinSi
13  FinFuncion
14
15  Proceso p5_ej3
16      Definir nro, val, k Como Entero
17      Dimension val[3]
18
19      Para k <- 1 Hasta 3 Con Paso 1
20          ingreso(nro)
21          val[k] <- calculo(nro)
22          Escribir "El nro ingresado fue: ", nro
23          Escribir "El número calculado fue: ", val[k]
24      FinPara
25  FinProceso
```

Parámetros formales

Parámetros actuales o reales

Tipos de parámetros: por valor y por referencia.

por valor: son solo parámetros de **entrada para el subalgoritmo**.
El parámetro formal recibe **una copia** del contenido de la variable o el resultado de una expresión al realizar la llamada o invocación del subprograma.
Las modificaciones que se realicen a esos valores en el subprograma **NO** se reflejarán fuera del mismo.
Si el parámetro formal es una **variable simple** y no se define el tipo de pasaje, se asume que es **por valor**.

Te doy una copia. El parámetro formal recibe únicamente una copia del contenido del parámetro actual.

Seguridad Total

Las modificaciones que el subalgoritmo le haga a ese valor **NO se reflejarán** en el programa original.
(Nota: Es el método por defecto si no se especifica en PSeInt).

Tipos de parámetros: por valor y por referencia.

por referencia: el parámetro formal como la variable utilizada al momento de invocar al subalgoritmo (parámetro actual) comparten la misma posición de memoria. Cualquier cambio que se realice al parámetro formal en el cuerpo del módulo o subalgoritmo, se refleja directamente en el respectivo parámetro actual.

Los arreglos siempre se consideran por referencia

Compartimos el original.
El parámetro formal y el actual comparten la misma posición de memoria. No hay copias.

Impacto Directo

Cualquier cambio realizado en el subalgoritmo **altera permanentemente** la variable **original** en el programa que lo invocó.
¡Es el mecanismo para devolver múltiples resultados!

Por Valor	Por Referencia
¿Qué se envía? Una **COPIA** del contenido.	**¿Qué se envía?** Acceso a la misma **POSICIÓN DE MEMORIA** original.
¿Si se modifica adentro, cambia afuera? **NO**.	**¿Si se modifica adentro, cambia afuera?** **SÍ** (se refleja directamente).
Tipos permitidos al invocar: Variables, Constantes (ej. 10), Expresiones (ej. 6+4).	**Tipos permitidos al invocar:** **SOLO** variables simples o arreglos.
Por defecto: Si no se especifica, PSeInt asume 'Por Valor'.	

Por Valor: Dibujar sobre la copia no afecta el original.

Por Referencia: Ambos abren el mismo cajón de la memoria.

```
1  SubProceso ingreso(nro Por Referencia)
2      Escribir "Ingrese un nro"
3      Leer nro
4  FinSubProceso
5
6  Funcion aux <- calculo(nro)
7      Definir aux Como Entero
8      Si nro < 0 Entonces
9          aux <- abs(nro)
10     Sino
11         aux <- nro ^ 2
12     FinSi
13 FinFuncion
14
15 Proceso p5_ej3
16     Definir nro, val, k Como Entero
17     Dimension val[3]
18
19     Para k <- 1 Hasta 3 Con Paso 1 Hacer
20         ingreso(nro)
21         val[k] <- calculo(nro)
22         Escribir "El nro ingresado fue: ", nro
23         Escribir "El número calculado fue: ", val[k]
24     FinPara
25 FinProceso
```

Se modifica si se modifica en el subalgoritmo, comparten espacio de memoria.

Solo hace una copia.

Actividad 9: Calcular el porcentaje de estudiantes regulares, libres y promocionados de una materia.

Como la regla dicta que una Función solo puede devolver exactamente **1 valor**, NO podemos resolver este problema usando una Función matemática tradicional. ¿Cómo lo resolvemos?

Calcular el porcentaje de estudiantes regulares, libres y promocionados de una materia.

¿Cuántos valores debe retornar? **3**

Como es más de 1, **NO puedo resolverlo con una función.**

¿Cómo lo resuelvo?

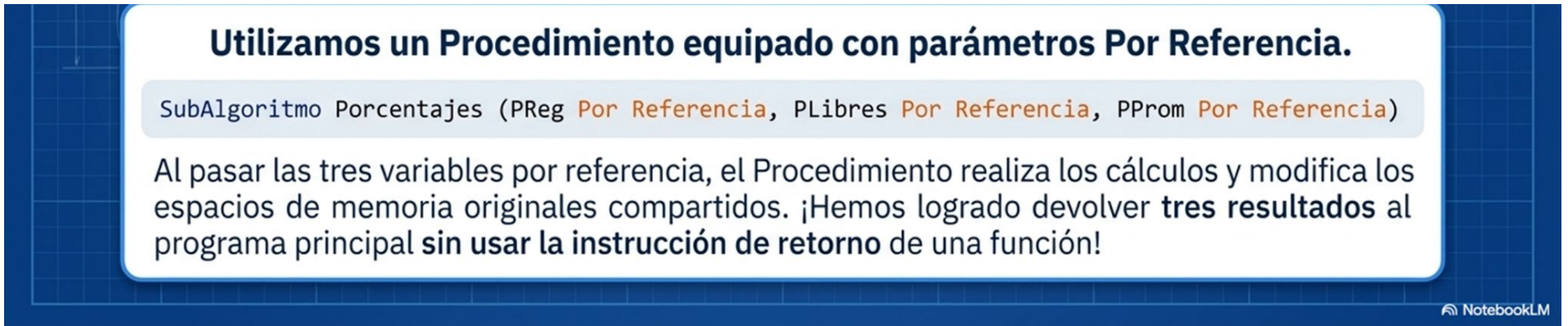
SubAlgoritmo Porcentajes (PReg **por referencia**, PLibres **por referencia**, PProm **por referencia**)

—---

—---

—---

FinSubalgoritmo



Calcular el porcentaje de estudiantes de la ingeniería en computación, del profesorado en computación y de la ingeniería en minas que cursan la materia.

¿Función o procedimiento?

Para los **parámetros formales** que fueron definidos:

por valor: los parámetros actuales pueden ser constantes (el valor **10**), variables (definidas en el ambiente del módulo invocante como **limsup**) o expresiones (como **6+4**).

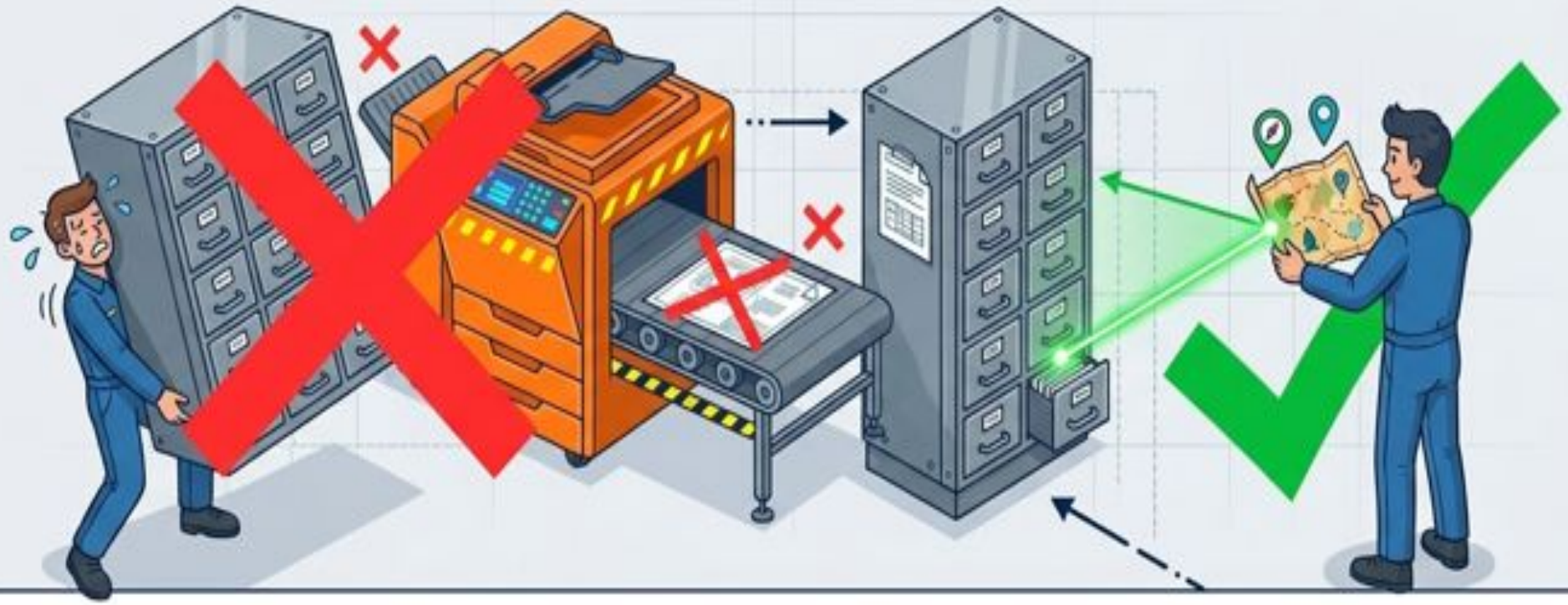
```
Proceso PromedioNotas
    Definir Notas como Real
    Definir Prom como Real
    Definir limsup como Entero
    Dimension Notas[10]
    limsup<-10

    IngresoNotas(10, Notas)
    IngresoNotas(limsup, Notas)
    IngresoNotas(6+4, Notas)
```

por referencia: los parametros actuales deben ser variables simples o **arreglos definidos en el ambiente del ḿodulo invocante**, pues el subalgoritmo devuelve sus resultados (como el arreglo **Notas**).

El parámetro formal y el actual comparten las posiciones de memoria asignadas a esas variables.

Arreglos como Parámetros



Los arreglos SIEMPRE se consideran pasados Por Referencia.

Por eficiencia, nunca se hace una copia de un arreglo completo. Si modificas una posición del arreglo **N** (parámetro formal) dentro de tu subalgoritmo, estás modificando el arreglo **Num** original en tu programa principal.

Arreglos como parámetros.

```
1   SubProceso Ingreso (N, ls)
2       Definir i como entero
3       Para i<-1 Hasta ls Con Paso 1 Hacer
4           Escribir "Ingrese un numero "
5           Leer N[i]
6       FinPara
7   FinSubProceso
8
9   Proceso ArregloComoParametro
10      Definir Num, i, limsup Como Entero
11      Dimension Num[10]
12      limsup<-4
13      Ingreso(Num, limsup)
14
15      Para i<-1 Hasta limsup Con Paso 1 Hacer
16          Escribir Num[i]
17      FinPara
18
19  FinProceso
20
```

N es un parámetro formal del procedimiento **Ingreso**.
N es un arreglo que se corresponde con el parámetro actual **Num**.

Los arreglos siempre se consideran por referencia

Si **N** **se modifica** **Num** también.

Modularizar el algoritmo que resuelve el enunciado:

Diseñar un algoritmo que permita registrar y procesar información sobre 34 estudiantes.

Para cada estudiante se debe almacenar:

- Número de registro (entero).
- Cantidad de materias que cursa (entero).
- Si posee o no beca de comedor (valor lógico: verdadero si posee beca, falso si no posee).

El algoritmo debe permitir:

a) Calcular el promedio de materias cursadas por los estudiantes.

b) Calcular el porcentaje de estudiantes que poseen beca de comedor.

c) Consultar los datos de un estudiante específico, ingresando su número de registro.

Versión 1: Descomposición en Tareas

T1: Definir `registros`, `materias` como arreglos de 34 posiciones de tipo entero y `tieneBeca` como arreglo de 34 posiciones de tipo lógico. Definir las variables `sumMaterias`, `cantBecados`, `i` y `regConsulta` de tipo entero; las variables `promedioMaterias` y `porcentajeBecas` de tipo real.

T2: Inicializar las variables `sumMaterias` en 0 y `cantBecados` en 0.

T3: Recorrer los arreglos desde la primera hasta la última posición, ingresando para cada estudiante su número de registro, su cantidad de materias y si posee beca de comedor.

T4: Recorrer el arreglo `materias` desde la posición 1 hasta la 34, **acumulando** todos los valores en la variable `sumMaterias`.

T5: Calcular el promedio de materias dividiendo `sumMaterias` por 34 y guardarlo en `promedioMaterias`.

T6: Mostrar el valor obtenido en `promedioMaterias`.

T7: Recorrer el arreglo `tieneBeca` desde la posición 1 hasta la 34. Si el valor es verdadero, incrementar (contar) la variable `cantBecados` en 1.

T8: Calcular el porcentaje de becas dividiendo `cantBecados` por 34 y multiplicando por 100. Guardar el resultado en `porcentajeBecas`.

T9: Mostrar el valor obtenido en `porcentajeBecas`.

T10: Solicitar el ingreso de un número de registro en la variable `regConsulta`.

T11: Recorrer el arreglo `registros`. Si el valor es igual a `regConsulta`, mostrar los datos almacenados en ese índice para los tres arreglos.

```
1  proceso GestionEstudiantes_Modula
2  Definir registros, materias, sumM
3  Definir promedioMaterias, porcent
4  Definir tieneBeca Como Logico
5  Dimension registros[34], materias
6
7  sumMaterias <- 0
8  cantBecados <- 0
9
10 Para i <- 1 Hasta 34 Hacer
11     Escribir "Estudiante nro
12     Leer registros[i], materi
13 FinPara
14
15 Para i <- 1 Hasta 34 Hacer
16     sumMaterias <- sumMate
17 FinPara
18 promedioMaterias <- sumMaterias /
19 Escribir "El promedio de materias
20
21 Para i <- 1 Hasta 34 Hacer
22     Si tieneBeca[i] = Verdadero E
23         cantBecados <- cantBecado
24     FinSi
25 FinPara
26 porcentajeBecas <- (cantBecados / 3
27 Escribir "El porcentaje de becados
28
29 Escribir "Ingrese registro a buscar:"
30 Leer regConsulta
31 Para i <- 1 Hasta 34 Hacer
32     Si registros[i] = regConsulta Ent
33         Escribir "Registro: ", regist
34         Escribir "Materias: ", materi
35         Escribir "Beca: ", tieneBeca[i]
36     FinSi
37 FinPara
38
39 Finproceso
```

T1: Definir `registros`, `materias` como arreglos de 34 posiciones de tipo entero y `tieneBeca` como arreglo de 34 posiciones de tipo lógico. Definir las variables `sumMaterias`, `cantBecados`, `i` y `regConsulta` de tipo entero; las variables `promedioMaterias` y `porcentajeBecas` de tipo real.

T2: Inicializar las variables `sumMaterias` en 0 y `cantBecados` en 0.

T3: Recorrer los arreglos desde la primera hasta la última posición, ingresando para cada estudiante de registro, su cantidad de materias y si posee beca de comedor.

T4: Recorrer el arreglo `materias` desde la posición 1 hasta la 34, **acumulando** todos los valores en la variable `sumMaterias`.

T5: Calcular el promedio de materias dividiendo `sumMaterias` por 34 y guardarlo en `promedioMat`

T6: Mostrar el valor obtenido en `promedioMaterias`.

T7: Recorrer el arreglo `tieneBeca` desde la posición 1 hasta la 34. Si el valor es (contar) la variable `cantBecados` en 1

T8: Calcular el porcentaje de becas dividiendo `cantBecados` por 34 y multiplicando por 100. Guardar e resultado en `porcentajeBecas`.

T9: Mostrar el valor obtenido en `porcentajeBecas`.

T10: Solicitar el ingreso de un número de registro en la varia

T11: Recorrer el arreglo `registros`. Si el valor es igual a `regConsulta` ese índice para los tres arreglos.

Versión 1: Descomposición en Tareas (Algoritmo Principal)

T1: Definir las variables `promedioMaterias` y `porcentajeBecas` de tipo real, los arreglos `registros` y `materias` de tipo entero de 34 posiciones y el arreglo `tieneBeca` de tipo lógico de 34 posiciones.

T2: Invocar al procedimiento `CargarDatos` pasando como parámetros los arreglos `registros`, `materias` y `tieneBeca` para realizar el ingreso de la información.

T3: Asignar a la variable `promedioMaterias` el valor de retorno de la función `CalcularPromedio` (pasando como parámetro el arreglo `materias`).

T4: Mostrar el valor almacenado en `promedioMaterias`.

T5: Asignar a la variable `porcentajeBecas` el valor de retorno de la función `CalcularPorcentaje` (pasando como parámetro el arreglo `tieneBeca`).

T6: Mostrar el valor almacenado en `porcentajeBecas`.

T7: Invocar al procedimiento `ConsultarEstudiante` pasando como parámetros los arreglos `registros`, `materias` y `tieneBeca.`

```
48  Proceso GestionEstudiantes_Modular
49
50  Definir registros, materias Como Entero
51  Definir tieneBeca Como Logico
52  Definir promedioMaterias, porcentajeBecas Como Real
53  Dimension registros[34], materias[34], tieneBeca[34]
54
55  CargarDatos(registros, materias, tieneBeca)
56  promedioMaterias <- ObtenerPromedio(materias)
57  Escribir "El promedio de materias es: ", promedioMaterias
58
59  porcentajeBecas <- ObtenerPorcentaje(tieneBeca)
60  Escribir "El porcentaje de becados es: ", porcentajeBecas, "%"
61
62  ConsultarEstudiante(registros, materias, tieneBeca)
63
64  FinProceso
```

```
48  Proceso GestionEstudiantes_Modular
49
50  Definir registr
51  Definir tieneBe
52  Definir promedi
53  Dimension regis
54
55  CargarDatos(reg
56  promedioMaterias
57  Escribir "El pro                                    as
58
59  porcentajeBecas
60  Escribir "El por                                    Becas, "%"
61
62  ConsultarEstudia
63
64  FinProceso
```

T1: Definir las variables `promedioMaterias` y `porcentajeBecas` de tipo real, los arreglos `registros` y `materias` de tipo entero de 34 posiciones y el arreglo `tieneBeca` de tipo lógico de 34 posiciones.

T2: Invocar al procedimiento `CargarDatos` pasando como parámetros los arreglos `registros`, `materias` y `tieneBeca` para realizar el ingreso de la información.

T3: Asignar a la variable `promedioMaterias` el valor de retorno de la función `CalcularPromedio` (pasando como parámetro el arreglo `materias`).

T4: Mostrar el valor almacenado en `promedioMaterias`.

T5: Asignar a la variable `porcentajeBecas` el valor de retorno de la función `CalcularPorcentaje` (pasando como parámetro el arreglo `tieneBeca`).

T6: Mostrar el valor almacenado en `porcentajeBecas`.

T7: Invocar al procedimiento `ConsultarEstudiante` pasando como parámetros los arreglos `registros`, `materias` y `tieneBeca`.

```
1   // --- SUBALGORITMOS ---
2   Subproceso CargarDatos(reg, mat, b)
3       Definir j Como Entero
4       Para j <- 1 Hasta 34 Hacer
5           Escribir "Estudiante ", j, ":"
6           Escribir "Ingrese numero de registro "
7           Leer reg[j]
8           Escribir "Ingrese cantidad de materias "
9           Leer mat[j]
10          Escribir "Tiene beca?"
11          Leer b[j]
12      FinPara
13  FinSubproceso
14
15  Subproceso prom <- ObtenerPromedio(mat)
16      Definir j, acum Como Entero
17      acum <- 0
18      Para j <- 1 Hasta 34 Hacer
19          acum <- acum + mat[j]
20      FinPara
21      prom <- acum / 34
22  FinSubproceso
23
```

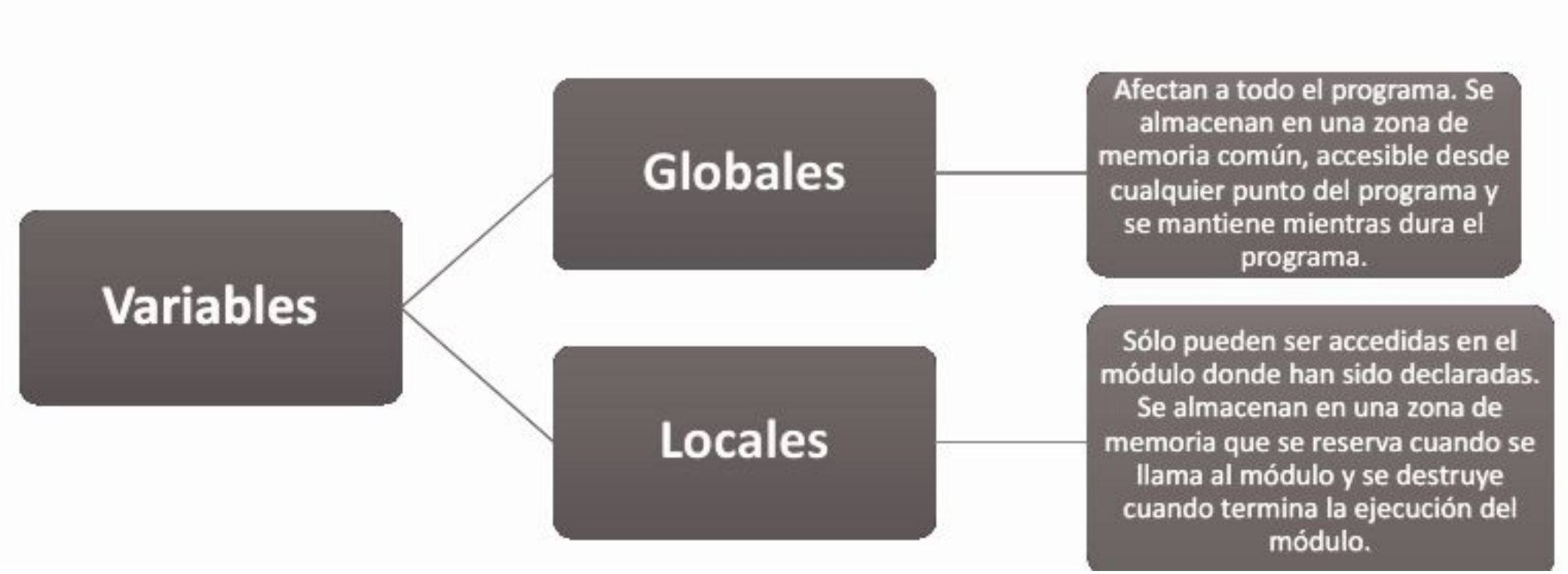
```
48  Proceso GestionEstudiantes_Modular
49
50  Definir registros, materias Como Entero
51  Definir tieneBeca Como Logico
52  Definir promedioMaterias, porcentajeBecas Como Real
53  Dimension registros[34], materias[34], tieneBeca[34]
54
55  CargarDatos(registros, materias, tieneBeca)
56  promedioMaterias <- ObtenerPromedio(materias)
57  Escribir "El promedio de materias es: ", promedioMaterias
58
59  porcentajeBecas <- ObtenerPorcentaje(tieneBeca)
60  Escribir "El porcentaje de becados es: ", porcentajeBecas, "%"
61
62  ConsultarEstudiante(registros, materias, tieneBeca)
63
64  FinProceso
```

```
24  Subproceso porc <- ObtenerPorcentaje(beca)
25      Definir j, conta Como Entero
26      conta <- 0
27      Para j <- 1 Hasta 34 Hacer
28          Si beca[j] = Verdadero Entonces
29              conta <- conta + 1
30          FinSi
31      FinPara
32      porc <- (conta / 34) * 100
33  FinSubproceso
35  Subproceso ConsultarEstudiante(reg, mat, b)
36      Definir j, busqueda Como Entero
37      Escribir "Ingrese registro a buscar:"
38      Leer busqueda
39      Para j <- 1 Hasta 34 Hacer
40          Si reg[j] = busqueda
41              Escribir "Registr
42              Escribir "Materia
43              Escribir "Beca: "
44          FinSi
45      FinPara
46  FinSubproceso
```

```
48  Proceso GestionEstudiantes_Modular
49
50  Definir registros, materias Como Entero
51  Definir tieneBeca Como Logico
52  Definir promedioMaterias, porcentajeBecas Como Real
53  Dimension registros[34], materias[34], tieneBeca[34]
54
55  CargarDatos(registros, materias, tieneBeca)
56  promedioMaterias <- ObtenerPromedio(materias)
57  Escribir "El promedio de materias es: ", promedioMaterias
58
59  porcentajeBecas <- ObtenerPorcentaje(tieneBeca)
60  Escribir "El porcentaje de becados es: ", porcentajeBecas, "%"
61
62  ConsultarEstudiante(registros, materias, tieneBeca)
63
64  FinProceso
```

Ámbito de las variables.

Es la zona del programa en que la misma está **definida** y por lo tanto puede ser **accedida y utilizada**.



Variables locales que están ocultas en el interior del módulo y son utilizadas, exclusivamente, por el módulo donde se definió.

Variables globales a las cuales se puede acceder desde cualquier módulo del programa.
Es decir, dos o más módulos pueden acceder a las mismas variables siempre que sean globales.

Actividad 3: Completar la tabla con el ambiente de cada subalgoritmo y del algoritmo principal.

```
1  SubProceso ingreso(nro Por Referencia)
2      Escribir "Ingrese un nro"
3      Leer nro
4  FinSubProceso
5
6  Funcion aux <- calculo(nro)
7      Definir aux Como Entero
8      Si nro < 0 Entonces
9          aux <- abs(nro)
10     Sino
11         aux <- nro ^ 2
12     FinSi
13 FinFuncion
14
15 Proceso p5_ej3
16     Definir nro, val, k Como Entero
17     Dimension val[3]
18
19     Para k <- 1 Hasta 3 Con Paso 1 Hacer
20         ingreso(nro)
21         val[k] <- calculo(nro)
22         Escribir "El nro ingresado fue: ", nro
23         Escribir "El número calculado fue: ", val[k]
24     FinPara
25 FinProceso
```

	AMBIENTE
ingreso	nro
calculo	
p5_ej3	

Teoría

Pseudocódigo: Subalgoritmos

- ✔ **Modularización.**
- ✔ **Diferencia entre Funciones y Procedimientos.**
- ✔ **Sintaxis en PSeInt de Funciones y Procedimientos.**
- ✔ **Definición e Invocación de Funciones y Procedimientos.**
- ✔ **Ejecución con subalgoritmos.**
- ✔ **Tipos de parámetros: por valor y por referencia.**
- ✔ **Arreglos como parámetros.**
- ✔ **Ámbitos de variables: locales o globales.**

Teoría

Pseudocódigo: Subalgoritmos

¡Ya podemos comenzar con el último Práctico de la materia!