Teoría

Pseudocódigo: Subalgoritmos

Resolución de Problemas y Algoritmos
Fundamentos de la programación

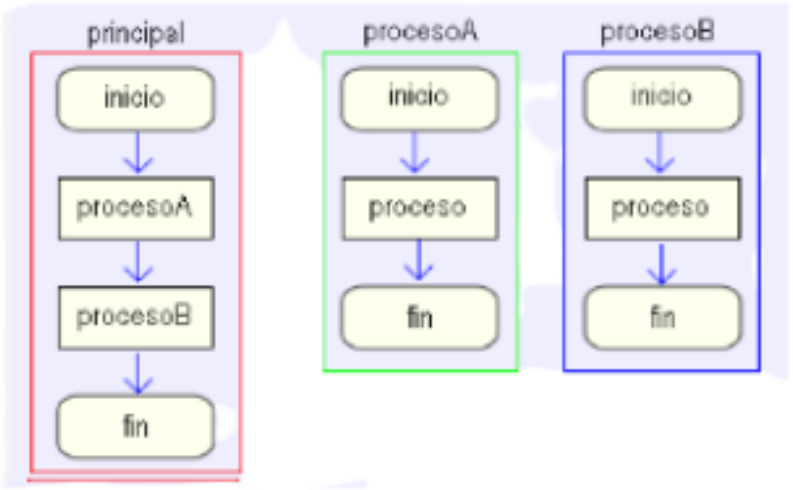
Ingeniería en Computación (TU y TFA)
Ingeniería en Minas (TU)
Profesorado en Ciencias de la Computación (TU y TFA)

Teoría

Pseudocódigo: Subalgoritmos

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

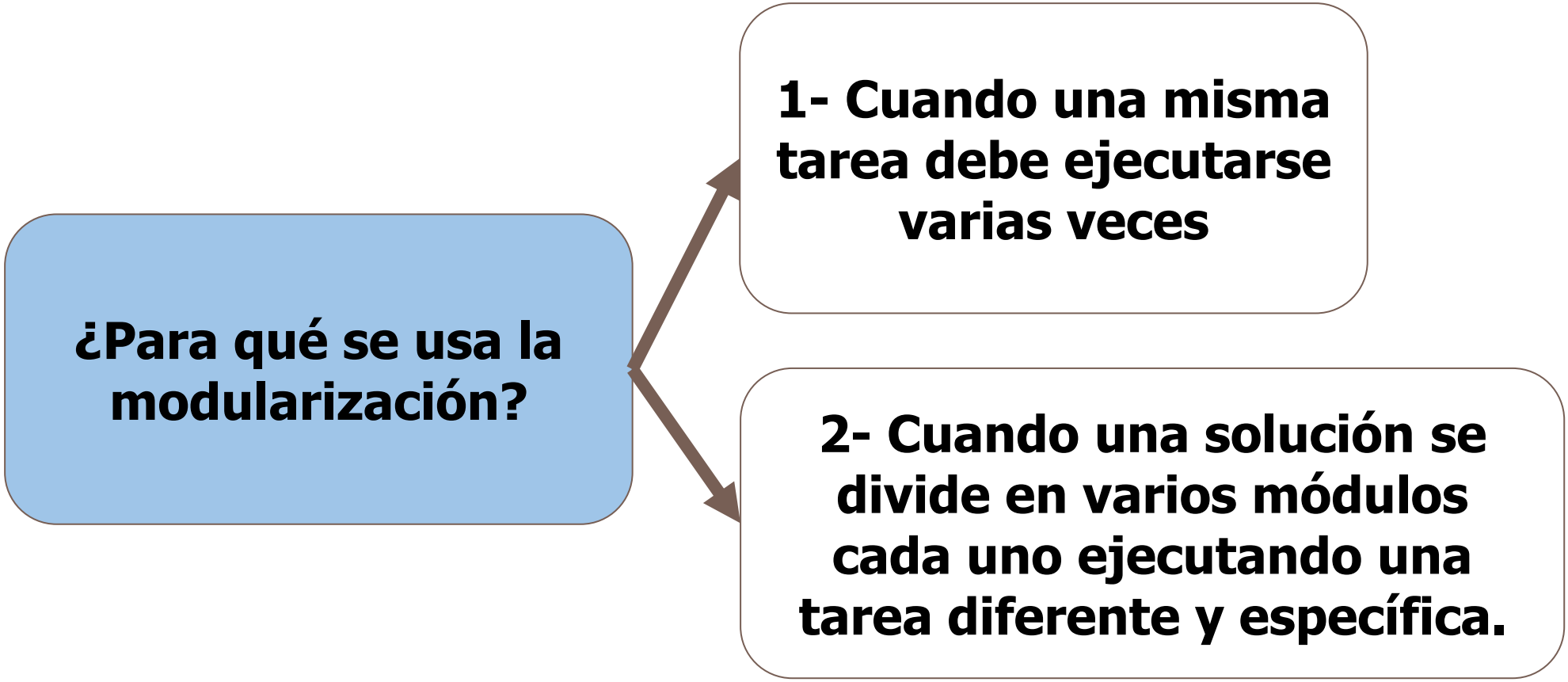
Modularización.



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).



Ventajas:

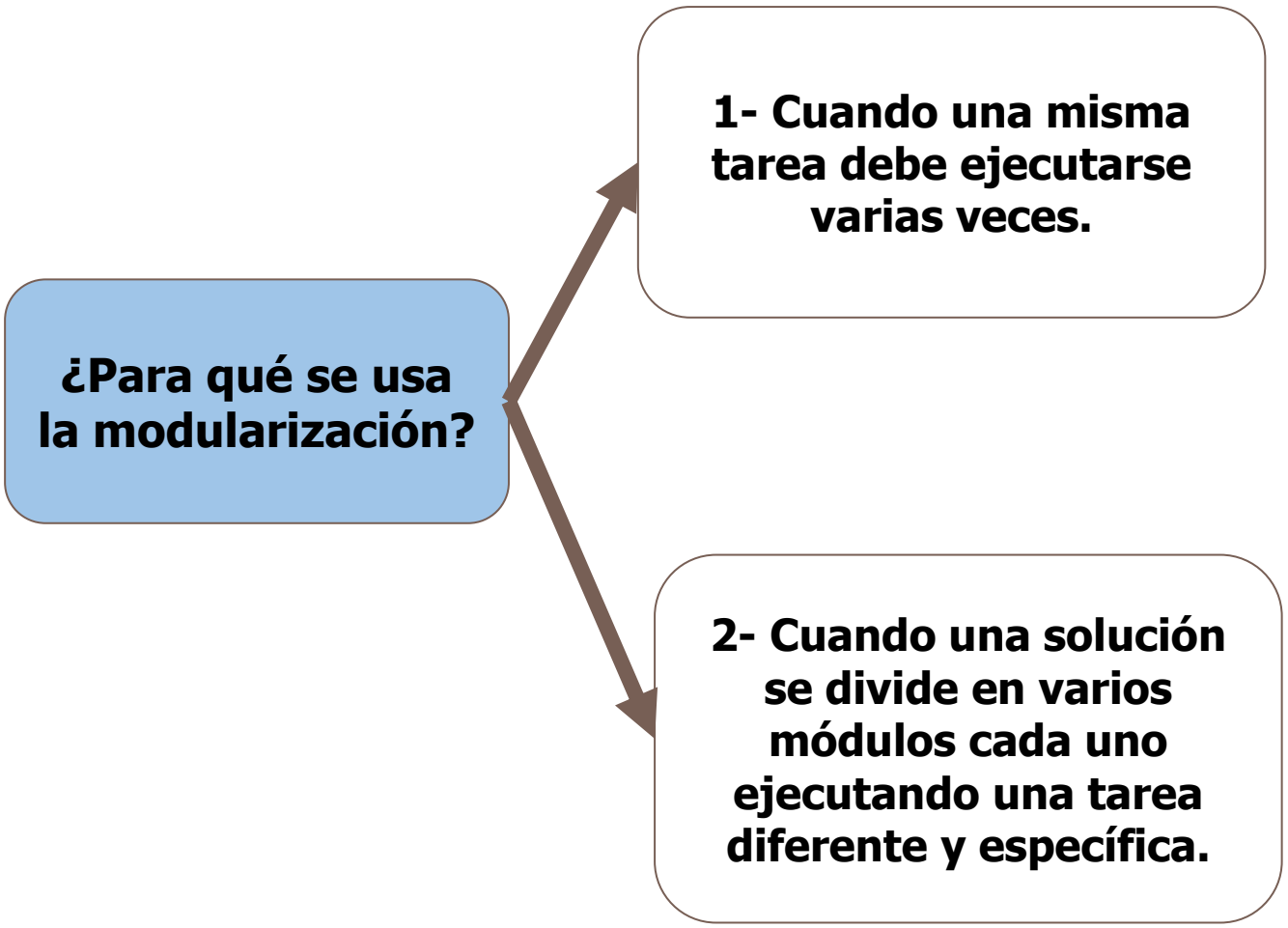
Aumenta la legibilidad y comprensión del programa.

Reduce el tiempo de desarrollo, aprovechando módulos ya existentes.

Permite la resolución del problema por varios programadores a la vez.

Facilita la depuración del programa.

Facilita el mantenimiento.



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.

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

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
2. …

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

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

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

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
```

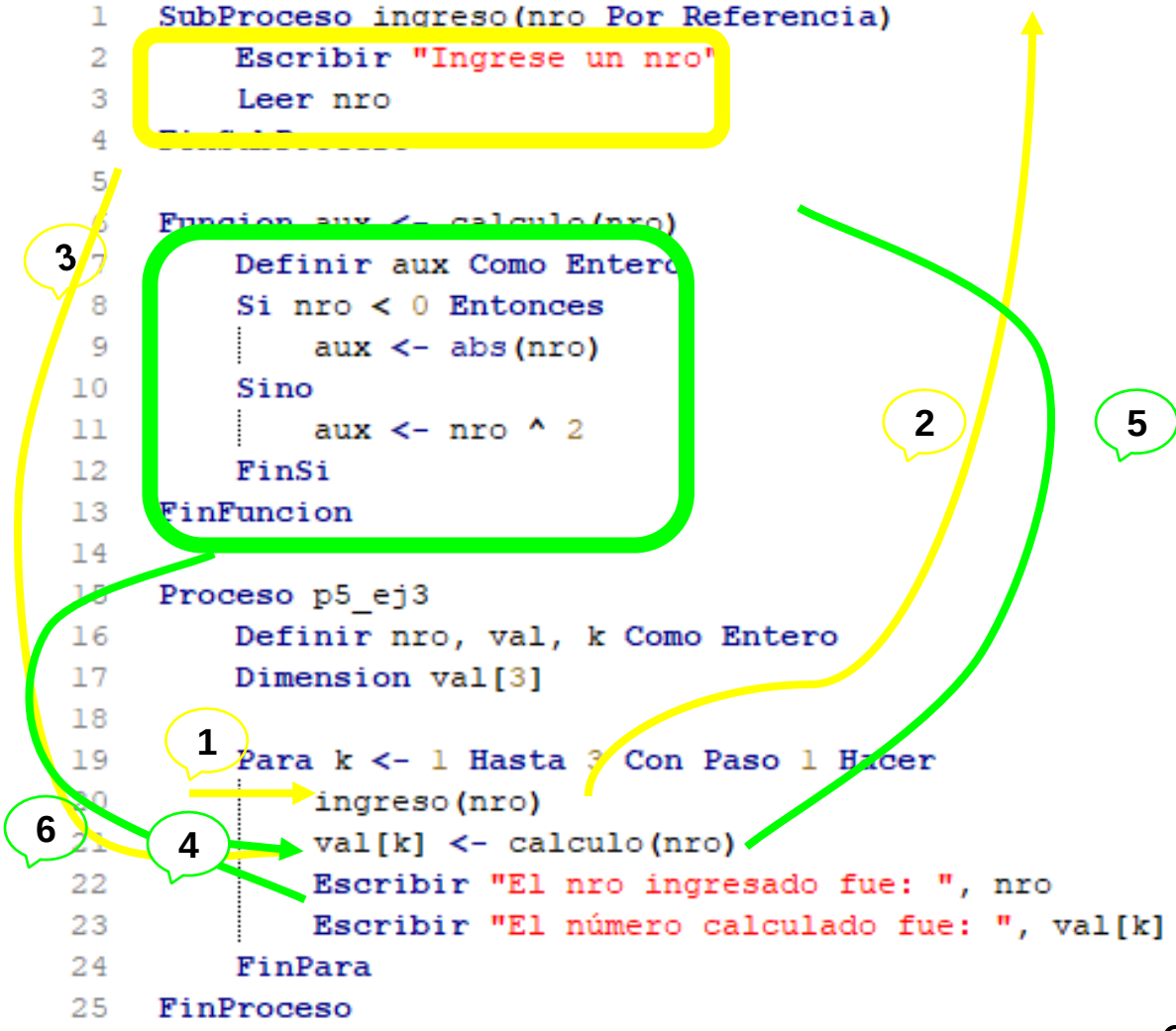
Para que las acciones descriptas en el subalgoritmo sean ejecutadas, es necesario que el mismo sea **invocado** desde el algoritmo principal o desde otro subalgoritmo.

```
1  SubProceso ingreso(nro Por Referencia)
2      Escribir "Ingrese un nro"
3      Leer nro
4  FinSubProceso
5
6  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: "
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   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
```


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.**

```
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 ps_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
```

Procedimiento

Función

Progama 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   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
```

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 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
```

Parámetros formales

Parámetros actuales o reales

PROCEDIMIENTOS

SIN RETORNO

- **SIN parámetros formales**
- **CON parámetros formales**

FUNCIONES

CON RETORNO

- **SIN parámetros formales**
- **CON parámetros formales**

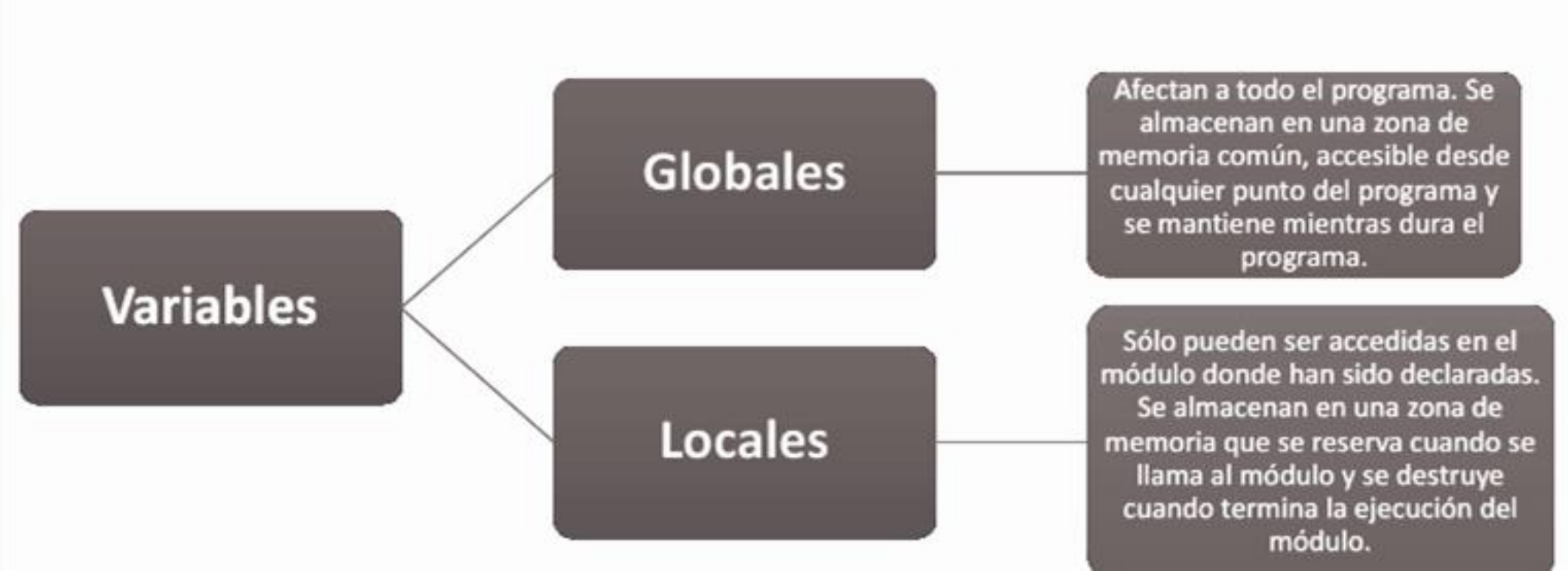
```
1  SubAlgoritmo ingreso ( nro por referencia)
2      Escribir "Ingrese un nro"
3      Leer Nro
4  FinSubalgoritmo
5  Subalgoritmo aux <- calculo (nro)
6  definir aux como entero
7      si nro < 0 entonces
8          aux ← abs(nro)
9      sino
10         aux ←    nro ↑ 2
11     FinSi
12 FinSubAlgoritmo
```

¿Cuántos parámetros se definen para un subalgoritmo?

PROCEDIMIENTO	**SIN parámetros**	SubProceso **ProcPromedioSinParam**() Definir Pr Como Real Pr<-(6 + 7)/2.0 escribir "El promedio sin parámetros es: ", Pr FinSubProceso
	CON parámetros	SubProceso **ProcPromedioConParam(n1 , n2)** Definir Pr Como Real Pr<-(n1 + n2)/2.0 escribir "El promedio con parametros es: ", Pr FinSubProceso
FUNCIONES	**SIN parámetros**	SubProceso **Pr <-FunPromedioSinParam()** Definir Pr Como Real Pr<-(6 + 7)/2.0 FinSubProceso
	CON parámetros	SubProceso **Pr <-PromedioFunConParam(n1 , n2)** Definir Pr Como Real Pr<-(n1 + n2)/2.0 FinSubProceso

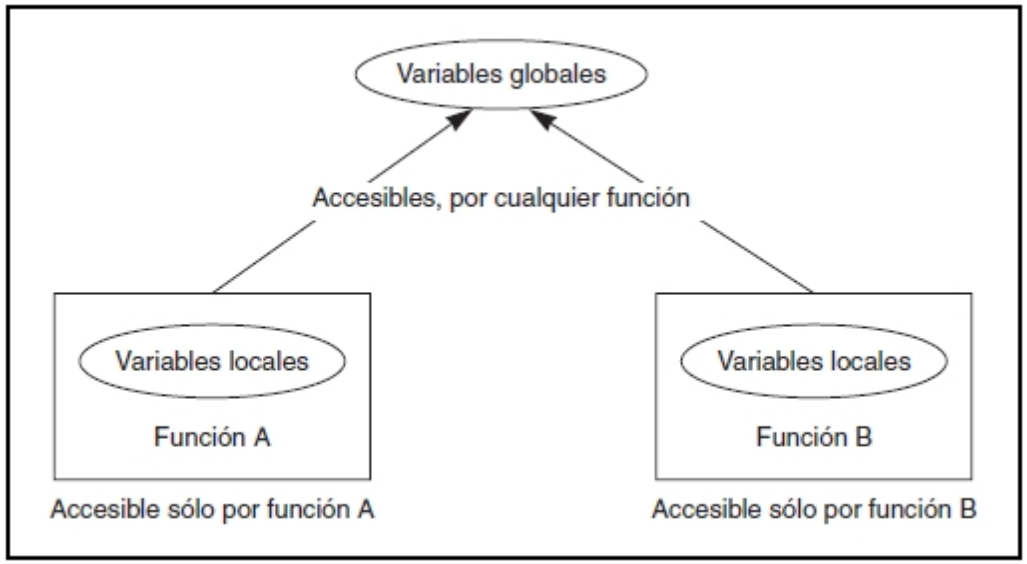
Ámbito de las variables.

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.
En la Figura se muestra la disposición de variables locales y globales en un programa.



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	

Tipos de parámetros: por valor y por referencia.

Los **parámetros formales** pueden ser definidos:

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**.

por referencia: es decir el subalgoritmo puede devolver resultados de la ejecución de sus acciones a quien lo invocó.
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, se refleja directamente en el respectivo parámetro actual.

Los arreglos siempre se consideran por referencia

```
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
```

Hace una copia y se modifica si se modifica en el subalgoritmo.

Solo hace una copia.

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

¿Cuántos valores debe retornar?

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

9. 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

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.

```
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.

Teoría

Pseudocódigo: Subalgoritmos

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

Teoría

Pseudocódigo: Subalgoritmos

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