

PROGRAMACIÓN I

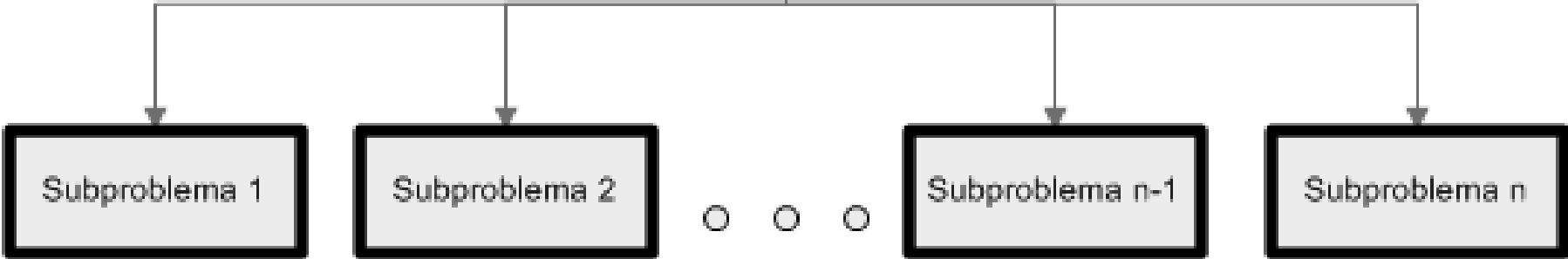
TEORÍA – CECILIA SANZ

Temas

- ✓ Modularización
- ✓ Ejemplos
- ✓ Comunicación entre módulos
- ✓ Ejemplos

MODULARIZACIÓN

***PROGRAMA
PRINCIPAL***



Actividad de Motivación

<https://www.youtube.com/watch?v=FIUxUMCNwnA>

Modularización

¿Qué observamos en los ejemplos del video?

- La tarea individual es tan importante como la grupal
- Cada uno hace una parte para resolver el problema
- El problema sólo se resuelve con el aporte de todos

Modularización

Los problemas del mundo real implican:

- Complejidad
- Extensión
- Modificaciones/Situaciones de cambio

Los tratamos de resolver con:

- Abstracción.
- Descomposición funcional.

Modularización

ABSTRACCIÓN

SALUDAR



LEVANTAR BRAZO
DERECHO

SACUDIR MANO
DERECHA

PRENDER LUZ



Modularización

ABSTRACCIÓN

DAR UN PASO

MOVER PIERNA
DERECHA

MOVER PIERNA
IZQUIERDA



Modularización

ABSTRACCIÓN

HABLAR

REPRODUCIR
SONIDO 1 - HOLA

REPRODUCIR SONIDO 2
– SOY DOBBY



Modularización

DESCOMPOSICIÓN FUNCIONAL



HABLAR

SALUDAR

**DAR UN
PASO**

**SENSAR
OBSTÁCULO**

**ESQUIVAR
OBSTÁCULO**

Modularización

Modularizar significa dividir un problema en partes **funcionalmente independientes**, que encapsulen operaciones y datos.



No se trata simplemente de subdividir el código de un sistema de software en bloques con un número de instrucciones dado.

Separar en funciones lógicas con datos propios y datos de comunicación perfectamente especificados.

Modularización – Abstracción

La descomposición tiene siempre un objetivo.

Se busca obtener:

Alta Cohesión: medida del grado de identificación de un módulo con una función concreta.

Bajo Acoplamiento: medida de la interacción de los módulos que constituyen un programa.

Modularización

DOBBY PRESENTARSE

HABLAR

SALUDAR

Como diseñamos con bajo acoplamiento y alta cohesión puedo reutilizar para crear nuevas funcionalidades.



Modularización – Abstracción

Cuando se descompone un problema en subproblemas, deben ser de forma tal que:

- Cada subproblema está en un mismo nivel de detalle.
- Cada subproblema puede resolverse lo más independientemente posible.
- Las soluciones de los subproblemas pueden combinarse para resolver el problema original.

Modularización – Descomposición

¿Qué son los Módulos?

Es un conjunto de instrucciones que cumplen una tarea específica bien definida, se comunican entre sí adecuadamente y cooperan para conseguir un objetivo común.



Cada módulo encapsula, acciones tareas o funciones



Hay que representar los objetos relevantes del problema a resolver.

Modularización

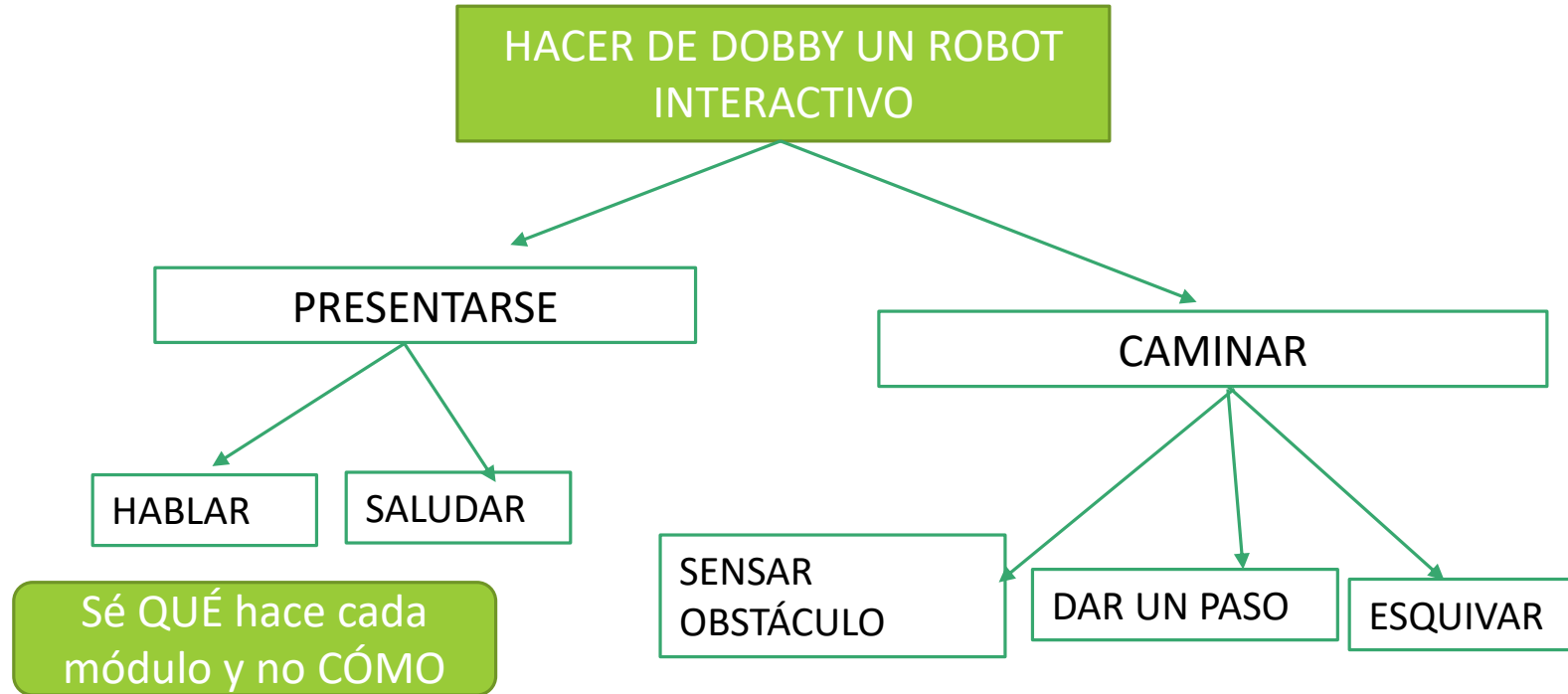
- ¿Es importante tener una buena metodología de trabajo?

TOP DOWN

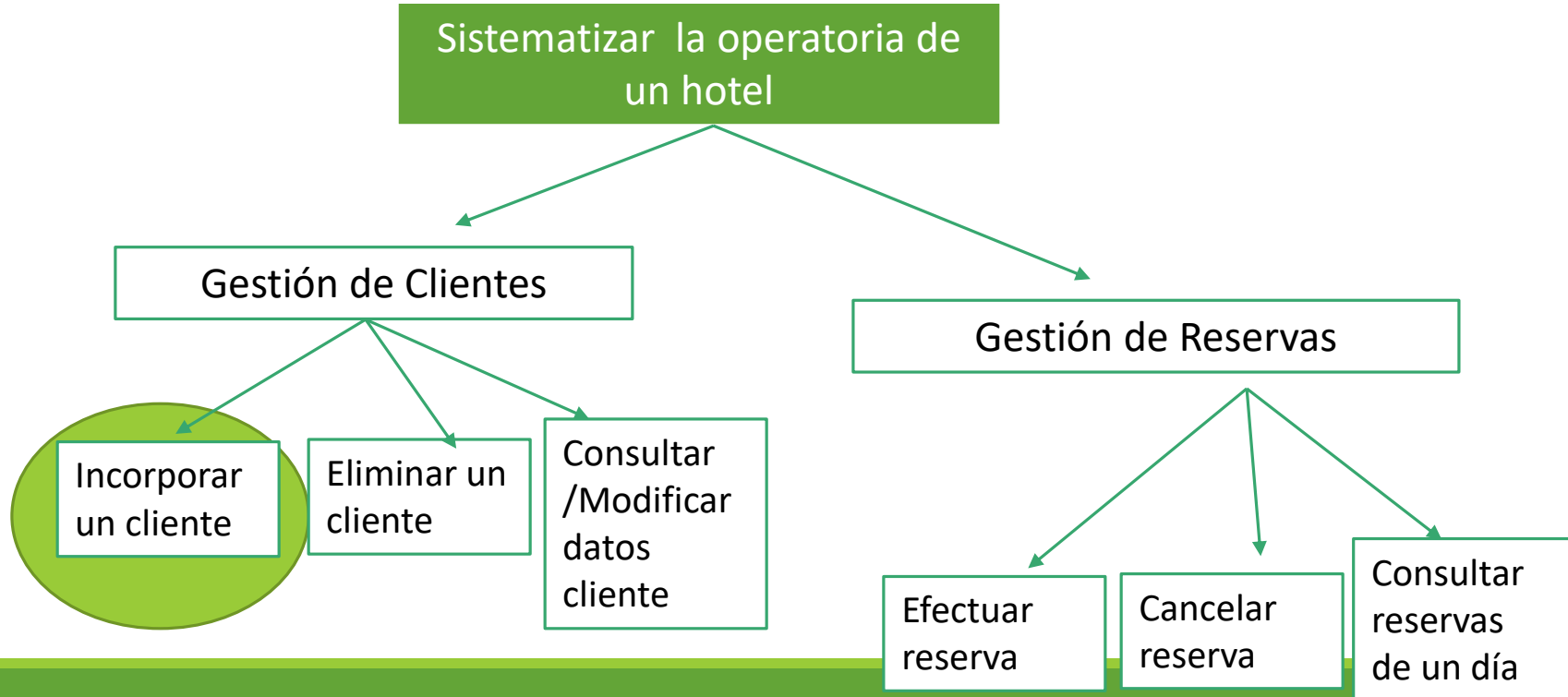
Ir de lo general a lo particular

Dividir ... conectar ... y verificar

Modularización – Ejemplo 1



Modularización – Ejemplo 2



VENTAJAS DE LA MODULARIZACIÓN

VENTAJAS DE LA MODULARIZACIÓN

Mayor productividad

Mayor legibilidad

Reusabilidad

**Facilidad de mantenimiento
correctivo**

**Facilidad de crecimiento del
sistema**

Modularización – Ventajas

Mayor productividad

Al dividir un sistema de software en módulos funcionalmente independientes, un equipo de desarrollo puede trabajar simultáneamente en varios módulos, incrementando la productividad (es decir reduciendo el tiempo de desarrollo global del sistema). Ejemplo.

Modularización – Ejemplo

Sistematizar la operatoria de un hotel

Gestión de Clientes

Incorporar un cliente

Eliminar un cliente

Consultar /Modificar datos cliente

Gestión de Reservas

Efectuar reserva

Cancelar reserva

Consultar reservas de un día



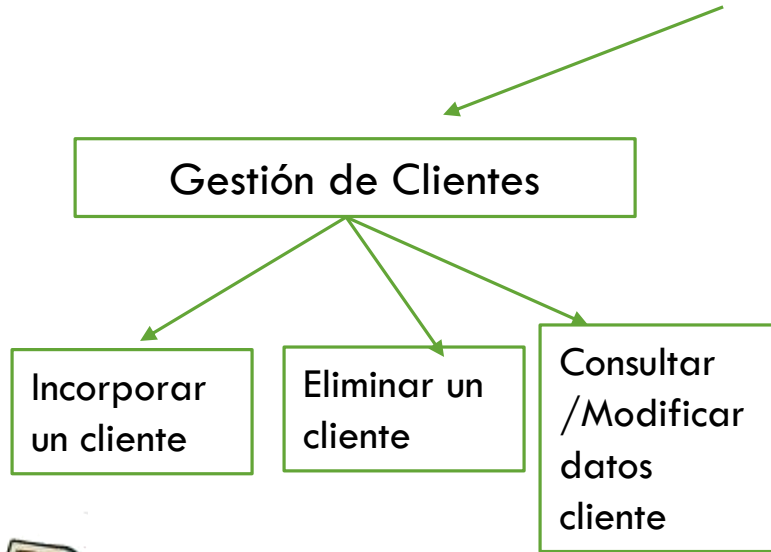
Modularización – Ventajas

Reusabilidad

Un objetivo fundamental de la Ingeniería de Software es la *reusabilidad*, es decir la posibilidad de utilizar repetidamente el producto de software desarrollado.

Naturalmente la **descomposición funcional** que ofrece la **modularización favorece el reuso**. Ejemplo.

Modularización – Ejemplo



Se puede utilizar
para cualquier otro
sistema



Modularización – Ventajas

Facilidad de Mantenimiento Correctivo

La división lógica de un sistema en módulos permite **aislar los errores** que se producen con mayor facilidad. Esto significa poder **corregir los errores** en menor tiempo y disminuye los costos de mantenimiento de los sistemas.

Ejemplo

Modularización – Ejemplo



No puedo eliminar
un usuario



**Eliminar un
usuario**

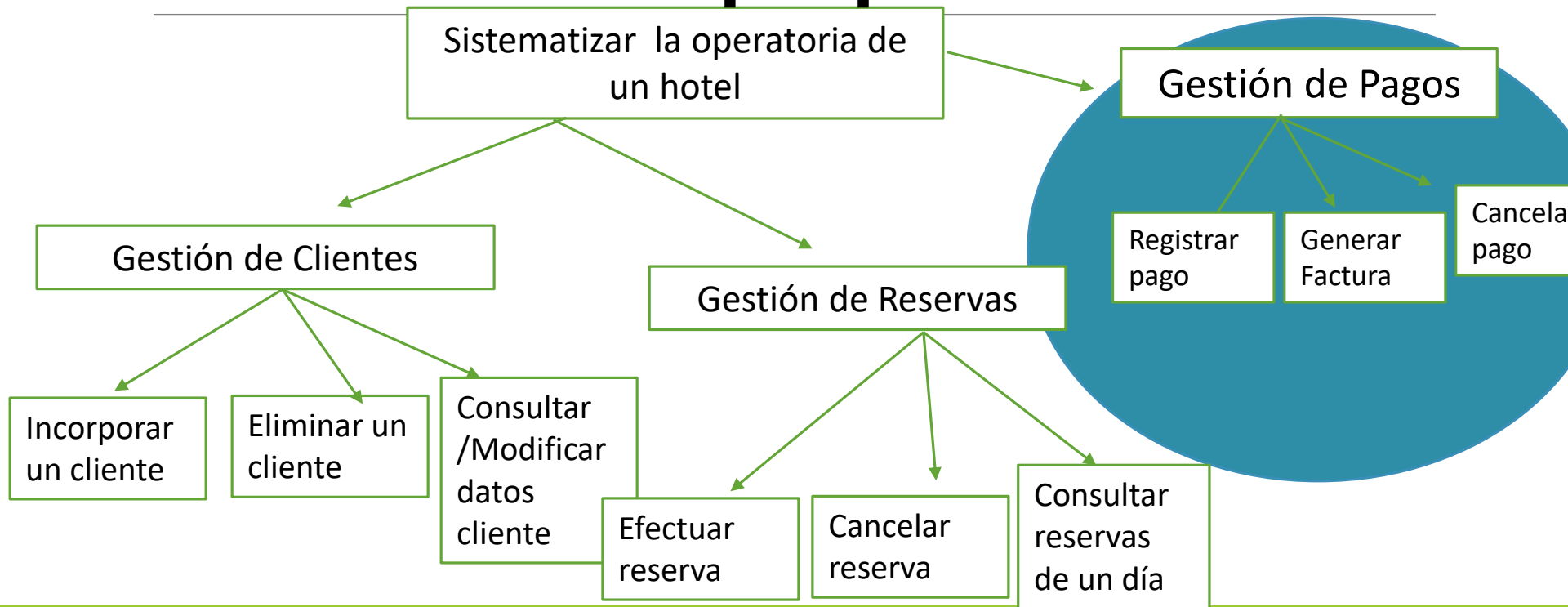


Modularización – Ventajas

Facilidad para el crecimiento del sistema

Los sistemas de software reales crecen (es decir aparecen con el tiempo nuevos requerimientos del usuario). La modularización permite **disminuir los riesgos y costos de incorporar nuevas prestaciones** a un sistema en funcionamiento. Ejemplo.

Modularización – Ejemplos



Modularización – Ventajas

Mayor Legibilidad

Un efecto de la modularización es una **mayor claridad para leer y comprender el código fuente**.

El ser humano maneja y comprende con mayor facilidad un número limitado de instrucciones directamente relacionadas.
Ejemplo.

Módulos en los lenguajes de programación

Modularización – Formas

Recursos de los lenguajes de programación para especificar la modularización

- **Subroutine** (ej.: fortran, perl)

- **Procedure**

- **Function**

(ej.:Pascal)

- **Package** (Ada)

etc.

Introducción a los módulos en Pascal

Módulos

Existen dos tipos de
módulos en Pascal

PROCEDIMIENTOS

FUNCIONES

Los módulos tienen:

- Un encabezado que permite mostrar qué hace
- Un cuerpo que oculta cómo resuelve su tarea
- Datos propios (Locales) y Datos compartidos

Los módulos para ejecutarse deben ser invocados o llamados

Modularización – Procedures

En R-INFO

```
programa uno
  procesos
    proceso auxiliar
      comenzar
    fin
  variables
    ....
  comenzar
  ...
fin
```

En Pascal

```
Program uno;
  procedure auxiliar;
  begin
  end;
  var
    ....
  begin
    ...
  end.
```

Program uno;

```
var
  ....
  procedure auxiliar;
  var
    ....
  begin
  end;
  begin
    ...
  end.
```

Modularización – Procedures

Conjunto de instrucciones que realizan una tarea específica y retorna 0, 1 o más valores.

¿Cómo se invoca?

```
Program uno;  
  Procedure auxiliar;  
    Var x: integer;  
    Begin  
      x:= 8;  
      x:= x * x;  
    End;  
  Begin  
    .....  
  End.
```

Procedimiento
sin parámetros.

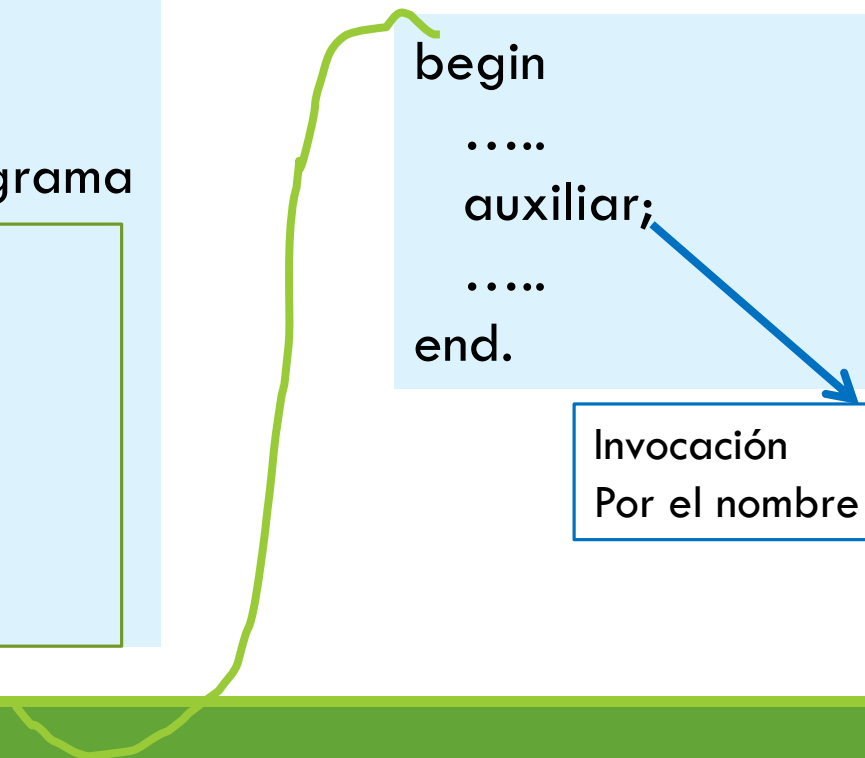
No devuelve nada.

Modularización – Procedures

```
Program uno;  
Var  
  // variables del programa  
procedure auxiliar;  
  Var  x:integer;  
  Begin  
    x:= 8;  
    x:= x * x;  
  End;
```

```
begin  
  ....  
  auxiliar;  
  ....  
end.
```

Invocación
Por el nombre



Modularización – Procedures

RESUMEN

1

Es un módulo que realiza tareas y puede devolver 0, 1 ó más valores.

2

Se invocan: escribiendo su nombre.

3

En general devuelve el resultado a través de PARAMETROS.

4

Respecto de las operaciones de lectura y escritura, no es aconsejable introducirlas como parte del módulo. Lo charlamos?

5

Permite parámetros de entrada y de entrada salida.

Modularización – Function

En R-INFO

No existe...

En Pascal

Program uno;

var

....

function auxiliar: tipo;

begin

.....

end;

Begin

...

End.

En Pascal

Program uno;

var

....

function auxiliar: tipo;

var

....

begin

.....

end;

Begin

...

End.

Modularización – Function

Conjunto de instrucciones que realizan una tarea específica y retorna 1 valor de tipo simple

```
function valor: integer;  
var  
  x:integer;  
begin  
  x:= 8;  
  valor:= x;  
end;
```

Function sin
parámetros.

Tipo que
devuelve.
Sólo simples

Devuelve el valor

Modularización – Function

```
Program uno;  
  Function auxiliar: integer;  
    var x:integer;  
  Begin  
    x:= 8*5;  
    auxiliar:= x;  
  End;  
Begin  
  ....  
End.
```



¿Cómo se invoca?

Modularización – Function

```
Program uno;  
Var res: integer;
```

```
Function auxiliar: integer;  
  var x:integer;  
Begin  
  x:= 8*5;  
  auxiliar:= x;  
End;
```

```
Begin
```

```
.....
```

```
res:= auxiliar;
```

```
.....
```

```
End.
```

Invocación a
la función

Por su nombre, asignando el
resultado a una variable del mismo
tipo que devuelve la función

Modularización – Function

```
Program uno;  
Var res: integer;
```

```
Function auxiliar: integer;  
  var x:integer;  
Begin  
  x:= 8*5;  
  auxiliar:= x;  
End;
```

```
Begin
```

```
.....
```

```
write ('El resultado es', auxiliar);
```

```
.....
```

```
End.
```

Invocación a
la función

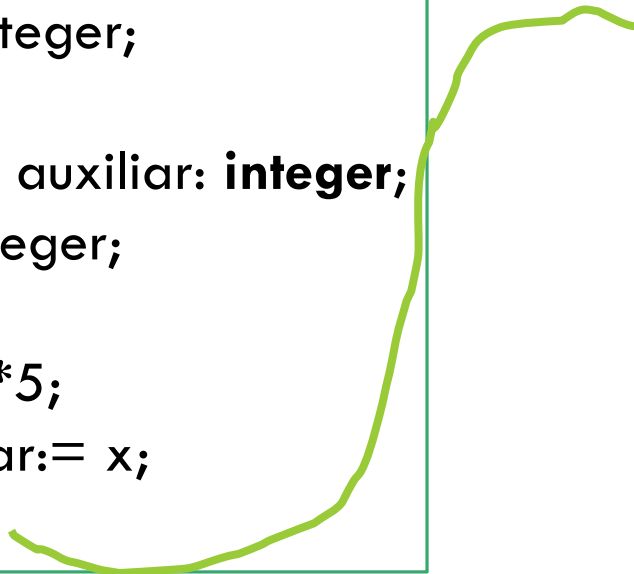


**Por su nombre, informando
su resultado.**

Modularización – Function

```
Program uno;  
Var res: integer;
```

```
Function auxiliar: integer;  
  var x:integer;  
Begin  
  x:= 8*5;  
  auxiliar:= x;  
End;
```




```
Begin
```

```
.....
```

```
  if (auxiliar = 5)  
  then write ('Dio cinco');
```

```
End.
```

Invocación a
la función




Por su nombre, en una
condición

Modularización – Function

```
Program uno;  
Var res: integer;
```

```
Function auxiliar: integer;  
  var x:integer;  
Begin  
  x:= 8*5;  
  auxiliar:= x;  
End;
```



```
Begin
```

```
...
```

```
while (auxiliar = 5) do  
  writeln('Quedó trabado');  
End.
```

Invocación a
la función



**Por su nombre, en una
condición**

Modularización – Function

RESUMEN

1

Es un módulo que realiza una única tarea y devuelve SIEMPRE un sólo valor de tipo simple.

2

Para devolver el resultado se asigna al nombre de la función como última instrucción.

3

Respecto de las operaciones de lectura y escritura, no es aconsejable introducirlas como parte del módulo. Lo charlamos?

4

Se pueden invocar: dentro de una condición (por ejemplo, en la condición de un if o de un while), o asignarla a una variable o dentro de un write.

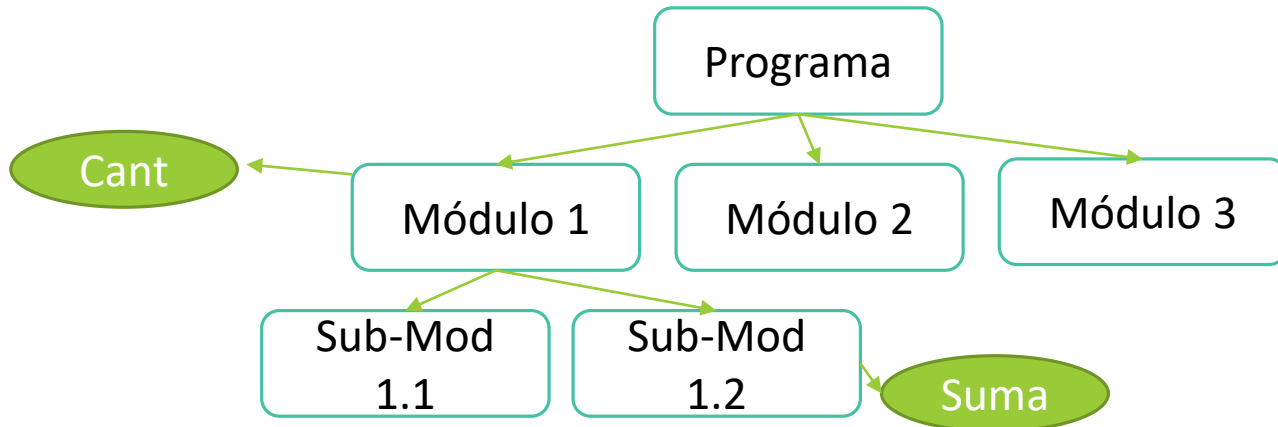
5

Pueden recibir sólo parámetros de entrada.

Concepto de Alcance de una variable

Concepto de Alcance de una variable

Alcance de una variable: es el contexto donde una variable pueden ser referenciada o nombrada y ésta es reconocida.



¿Cuál es el alcance de la variable Cant y cuál es el de la variable Suma?

Modularización – Alcance de variables

Program dos;

Var

a, b: integer;

procedure prueba;

var

x: integer;

Begin

x:= 9;

write (x);

End;

Begin

a:= 8;

b:= a*2;

End.

¿Donde se pueden utilizar a y b?

¿Donde se puede
utilizar x?

¿Qué pasa si dentro
de prueba se declara a: integer?

¿Qué pasa si dentro
de prueba se declara b: char?

Modularización – Alcance de variables

```
Program dos;
```

```
Var
```

```
a, b: integer;
```

```
procedure prueba;
```

```
var
```

```
Begin
```

```
x:= 9;
```

```
write (x);
```

```
End;
```

```
Begin
```

```
.....
```

```
End.
```

1. Se fija si es variable local

2. Si no es var. local, entonces se fija si es un parámetro.

3. Si no es var. local, y no es parámetro, entonces se fija si es var. global.

Modularización

Variable global: su declaración se hace en la sección de declaración del programa principal, es decir fuera de todos los módulos del programa y podrá ser usada en el programa y en todos los módulos del mismo.

Variable local: su declaración se hace en un módulo particular y sólo podrá ser usada por ese módulo. Si este módulo contiene a su vez otros módulos, entonces esa variable puede ser también usada por todos los módulos interiores.

Analicemos el siguiente ejemplo:

Program Ejemplo;

Var y, j: integer;

procedure prueba;

var x: integer;

procedure otro;

var z: integer;

begin

read (z);

z := z div 10;

write (x);

end;

Begin

x:= 9;

write (x);

x := x * j;

otro;

End;

Begin

j:=10; y:= j*2;

prueba;

End.

Modularización – ¿Qué imprime?

Program dos;

```
Var x : integer;  
procedure prueba;  
  var x: integer;  
  Begin  
    x:= 9;  
    write (x);  
  End;  
Begin  
  x:= 8;  
  prueba;  
  Writeln(x);  
End.
```

```
Program dos;  
Var x : integer;  
procedure prueba;  
  Begin  
    write (x);  
  End;  
  
Begin  
  x:=8;  
  prueba;  
End.
```

```
Program dos;  
Var x: integer;  
procedure prueba;  
  Begin  
    x:= 9;  
    write (x);  
  End;  
Begin  
  x:=8;  
  prueba;  
  writeln(x);  
End.
```

Resumen de aspectos importantes

Diferencias entre procedimiento y función

- ¿Dónde vuelve el flujo de control del programa una vez ejecutado el módulo?
- ¿Cómo se invocan?
- ¿Qué tipos de parámetros aceptan?
- Cuántos valores devuelven como mínimo?
- Operaciones que se pueden realizar en cada uno.