

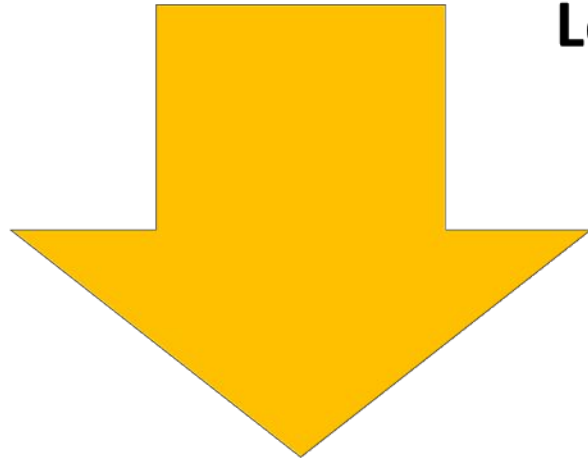
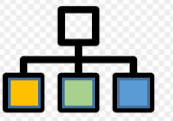
Conceptos de Algoritmos Datos y Programas

CADP – Temas de la clase de hoy



- Modularización
- Alcance de variables

CADP – MODULARIZACION

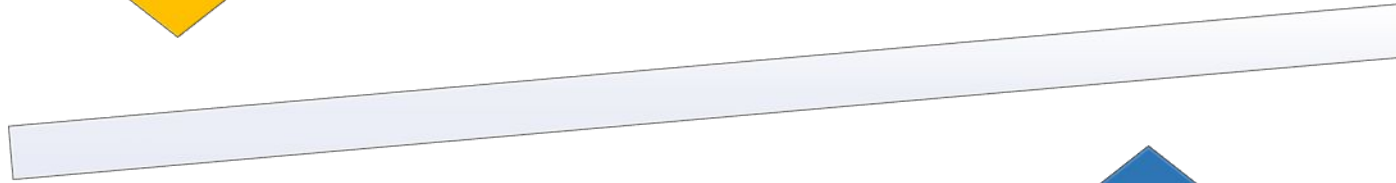


Los problemas del mundo real implican:

Complejidad

Extensión

Modificaciones

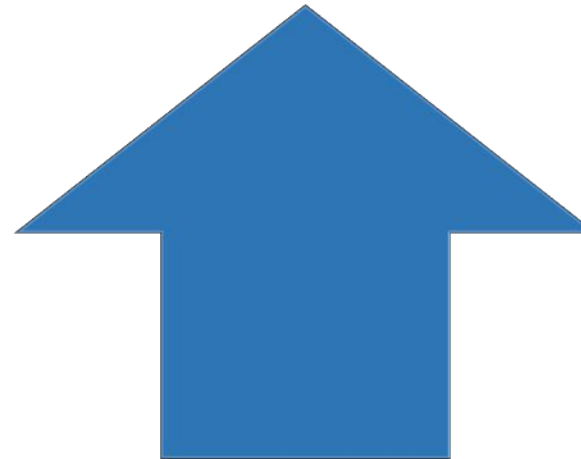


Los tratamos de resolver con:

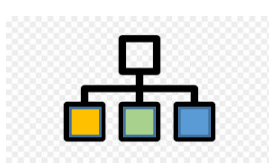
Abstracción

Descomposición

Independencia Funcional



CADP – MODULARIZACION



MODULARIZAR

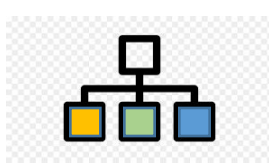
Significa dividir un problema en partes funcionalmente independientes, que encapsulen operaciones y datos.



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.



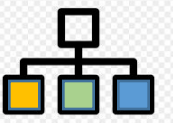
RECORDAR



Cada subproblema está en un mismo nivel de detalle.

Cada subproblema puede resolverse independientemente.

Las soluciones de los subproblemas puede combinarse para resolver el problema original.



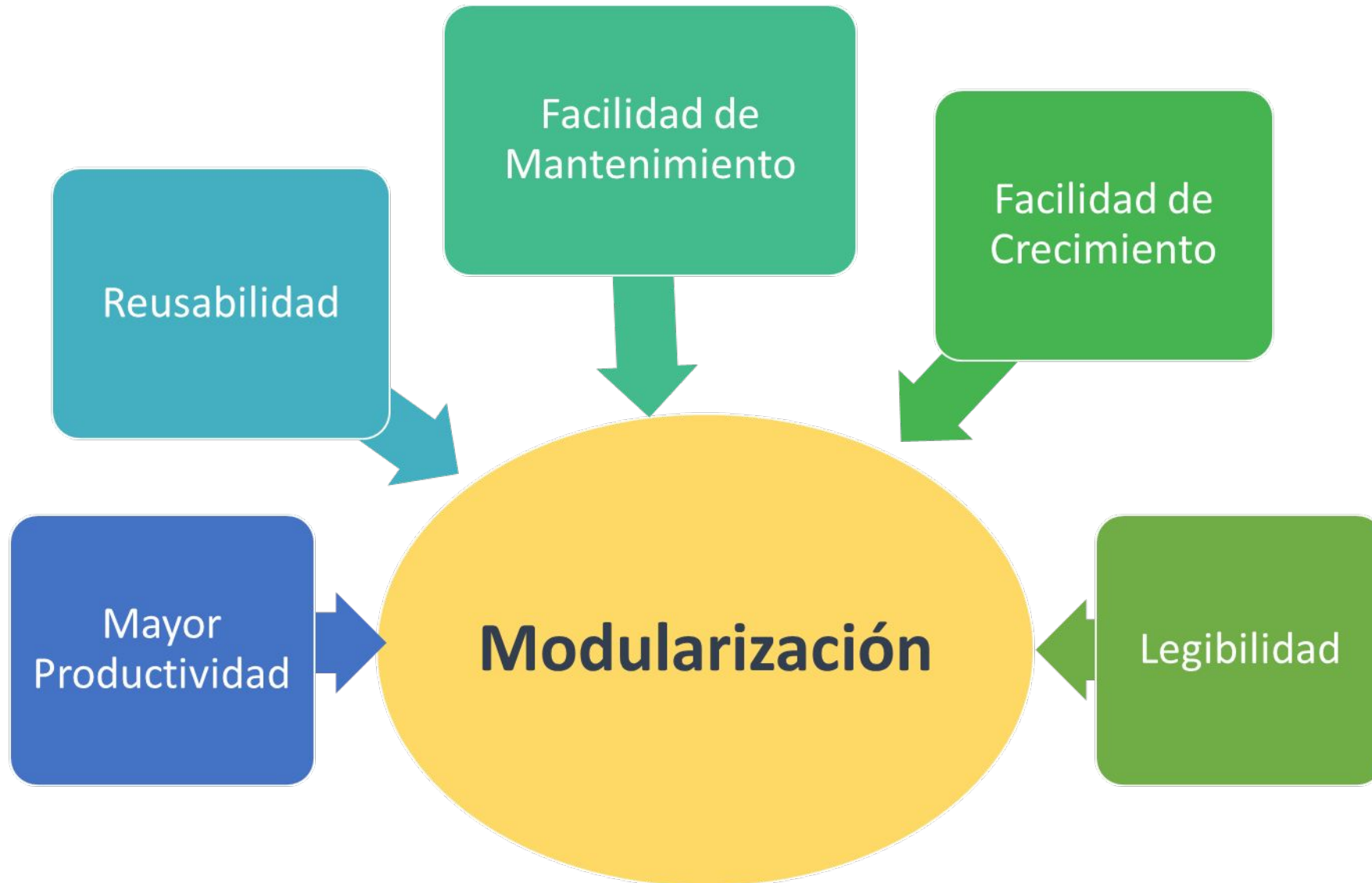
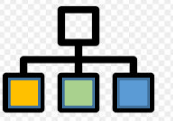
MODULO

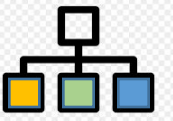
Tarea específica bien definida se comunican entre sí adecuadamente y cooperan para conseguir un objetivo común.

Encapsula acciones tareas o funciones.

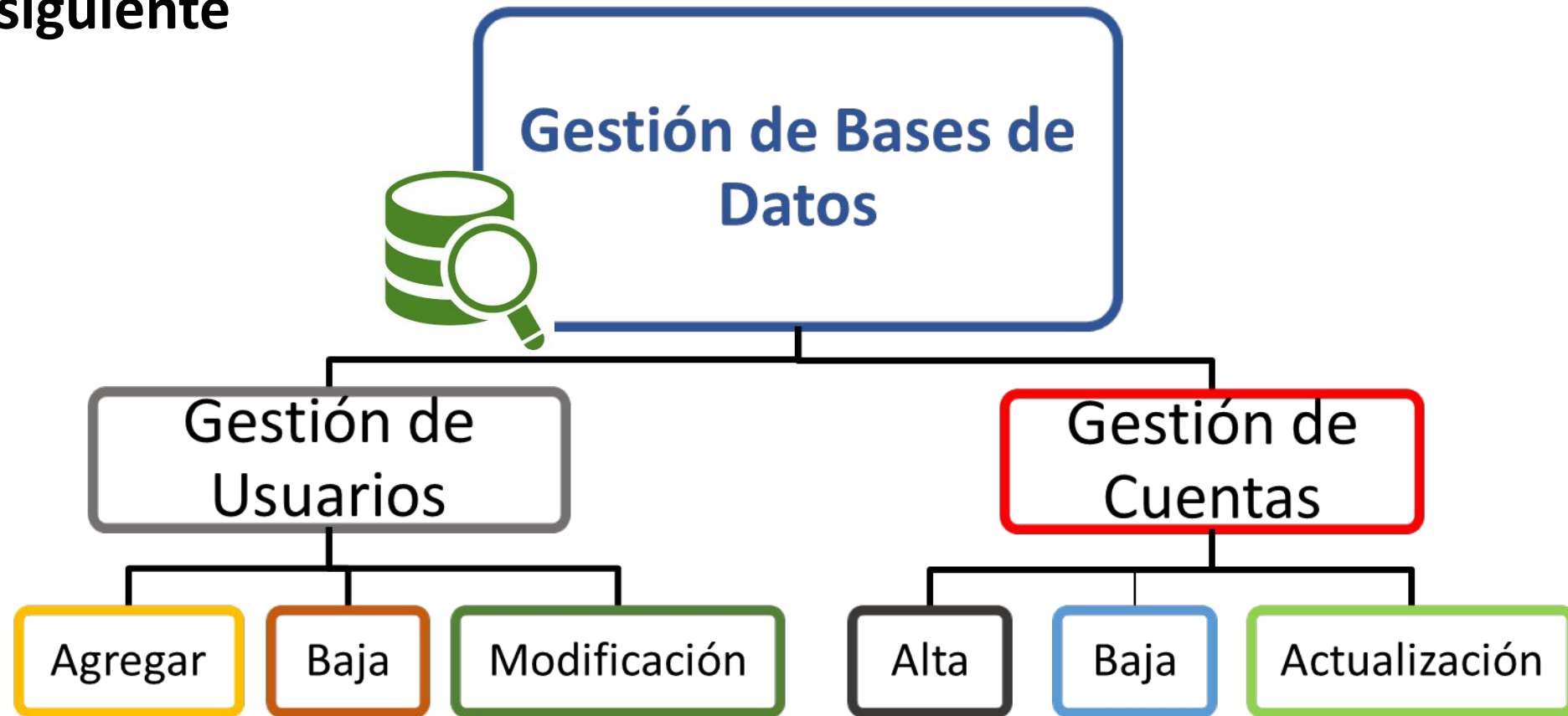
En ellos se pueden representar los objetivos relevantes del problema a resolver.

Existen diferentes metodologías para usarlos en los programas en particular nosotros usaremos la **METODOLOGIA TOP-DOWN**



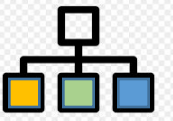


Supongamos que tenemos que resolver el siguiente proyecto



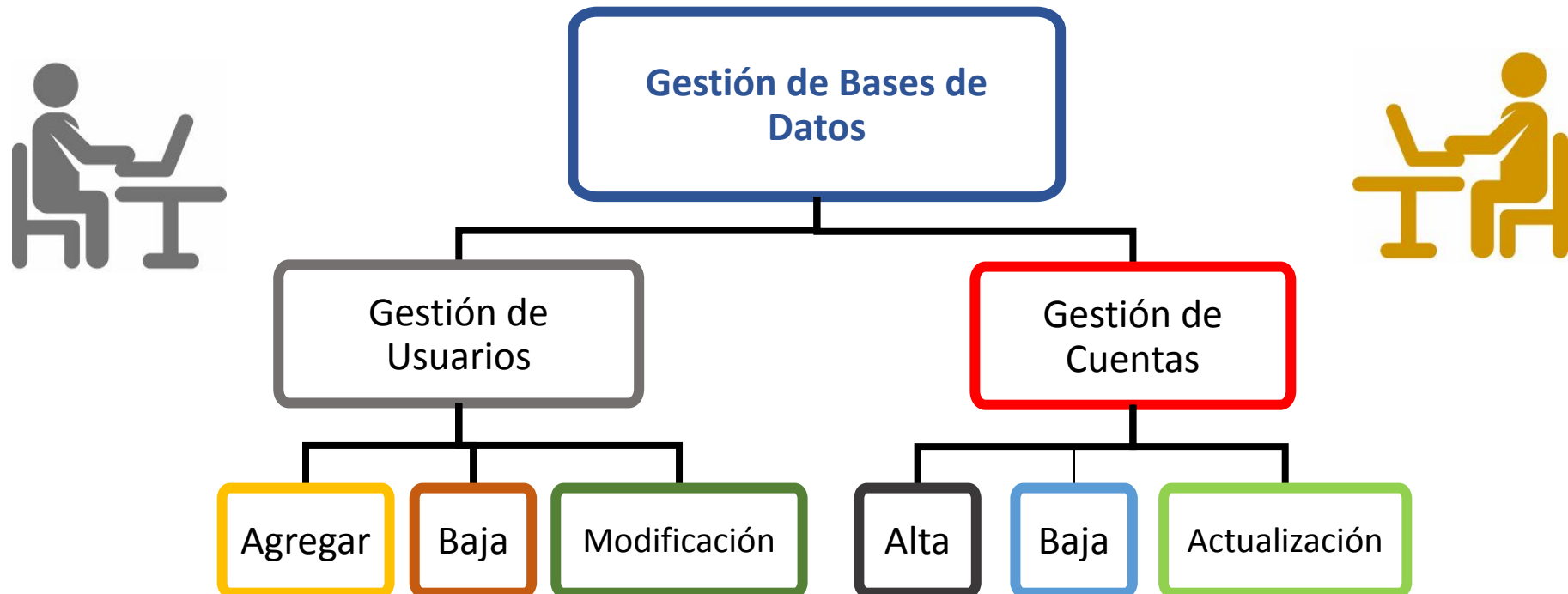
CADP – MODULARIZACION

VENTAJAS



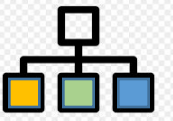
Al dividir un sistema de software en módulos funcionalmente independientes, un equipo de desarrollo puede trabajar simultáneamente en varios módulos, incrementando la productividad (es decir reduciendo el tiempo de desarrollo global del sistema).

**Mayor
Productividad**



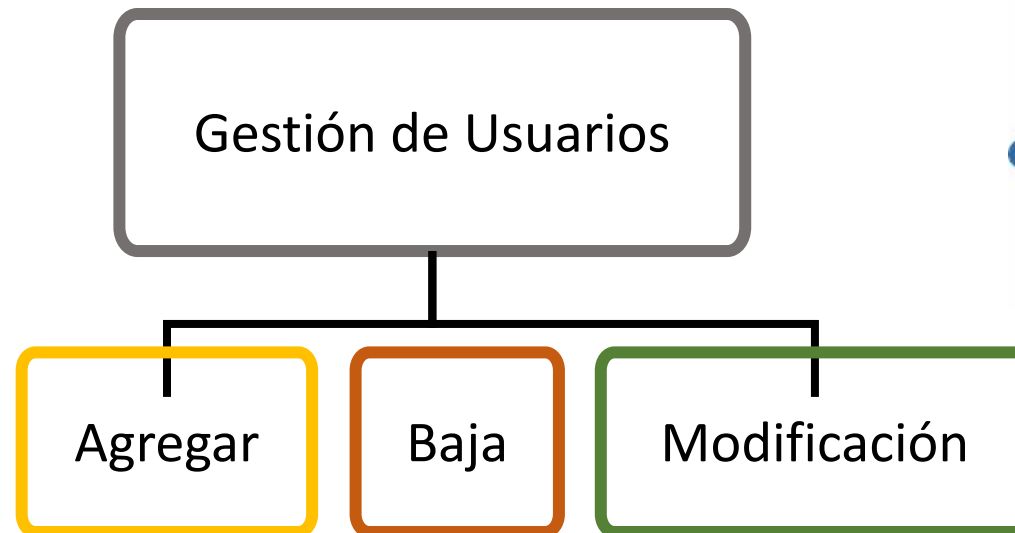
CADP – MODULARIZACION

VENTAJAS



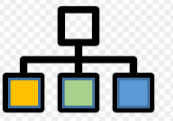
Un objetivo fundamental de la Ingeniería de Software es la reusabilidad, es decir la posibilidad de utilizar repetidamente el producto de software desarrollado. Naturalmente la descomposición funcional que ofrece la modularización favorece el reuso.

Reusabilidad



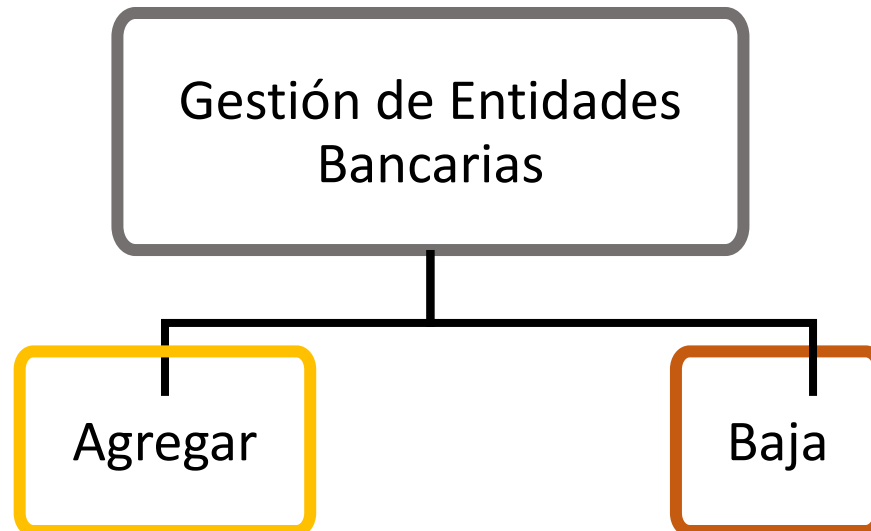
CADP – MODULARIZACION

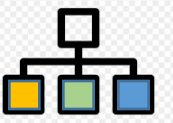
VENTAJAS



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.

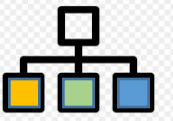
Facilidad de crecimiento





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.

Legibilidad



Fortran

- Subroutine

Modula

- Module

Ada, Pascal, C

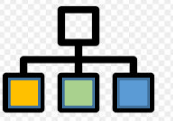
- Procedure/Function

Orientado a objetos

- Class

CADP – MODULARIZACION

PROCEDIMIENTOS

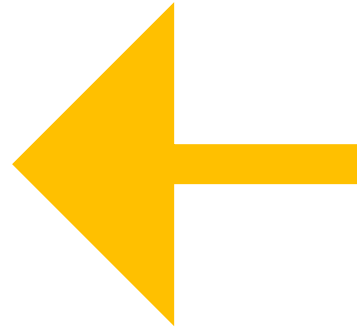


Programa nombre
areas

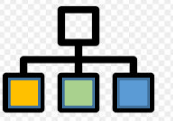
Procesos

proceso nombre
variables
comenzar
fin

variables
comenzar
fin



Cómo son?
Cómo se declaran?
Cómo se usan?



PROCEDIMIENTO

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

```
procedure nombre;
```

```
var
```

```
....
```

```
begin
```

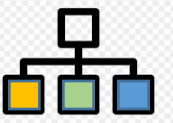
```
....
```

```
end;
```

Variables locales

Código del
procedimiento

Cómo se invocan?



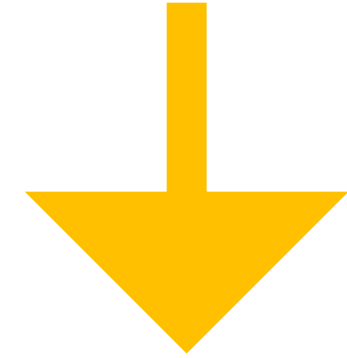
INVOCACION

```
Program uno;  
Const  
    ....  
Type  
    ....  
procedure auxiliar;  
Var  
    x:integer;  
begin  
    x:=8;  
end;
```

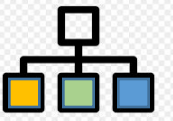
```
Var  
    ....  
Begin  
    auxiliar;  
End.
```



Por su nombre



**Existe otra forma
de modularizar
FUNCIONES**



FUNCION

Conjunto de instrucciones que realizan una tarea especifica y retorna un único valor de tipo simple.

```
Function nombre :tipo;
```

```
var
```

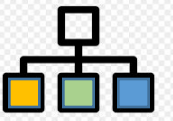
```
....
```

```
begin
```

```
....
```

```
end;
```

Qué diferencias se ven
con los procedimientos?



CARACTERISTICAS

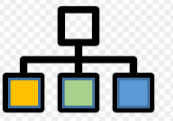
```
Function nombre() :tipo;  
var  
    ....  
begin  
    ....  
    nombre := valor a retornar;  
end;
```

Nombre de la función

Debe ser de tipo simple

Variables locales

Valor que devuelve la función



CARACTERISTICAS

```
program uno;
```

```
Function auxiliar: real;
```

```
Var
```

```
    x, y, cociente :real;
```

```
begin
```

```
    x:= 10;
```

```
    y:= 4;
```

```
    cociente:= x/y;
```

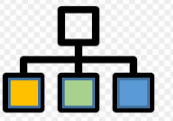
```
    auxiliar:= cociente;
```

```
end;
```

El valor que se calcula
(cociente) debe ser el mismo
tipo al que devuelve la función

*Cómo se
invoca?*

La asignación del nombre de la
función a la variable que retorna
debe ser la última instrucción.



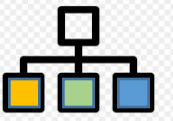
INVOCACION POR SU NOMBRE

Invocación usando variable

El resultado
se asigna a
una variable
del mismo
tipo que
devuelve la
función.

```
program uno;  
Function auxiliar: real;  
Var  
    x, y, cociente:real;  
  
begin  
    x:= 10;  
    y:= 4;  
    cociente:= x/y;  
    auxiliar:= cociente;  
end;  
Var  
    aux:real;  
begin  
    aux:= auxiliar;  
    write (aux);  
end.
```

El retorno de la
función es a la
misma línea de
invocación



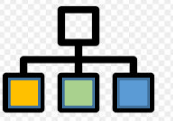
INVOCACION POR SU NOMBRE

Invocación en un while/if

El resultado
se asigna a
una variable
del mismo
tipo que
devuelve la
función.

```
program uno;  
Function auxiliar: real;  
Var  
    x, y, cociente:real;  
  
begin  
    x:= 10;  
    y:= 4;  
    cociente:= x/y;  
    auxiliar:= cociente;  
end;  
Var  
    aux:real;  
begin  
    while (auxiliar = 5.5) do  
        if (auxiliar = 5.5) then  
end.
```

El retorno de la
función es a la
misma línea de
invocación



INVOCACION POR SU NOMBRE

Invocación en un write

El resultado
se puede
mostrar en
una
sentencia
write.

```
program uno;
```

```
Function auxiliar: real;
```

```
Var
```

```
  x, y, cociente:real;
```

```
begin
```

```
  x:= 10;
```

```
  y:= 4;
```

```
  cociente:= x/y;
```

```
  auxiliar:= cociente;
```

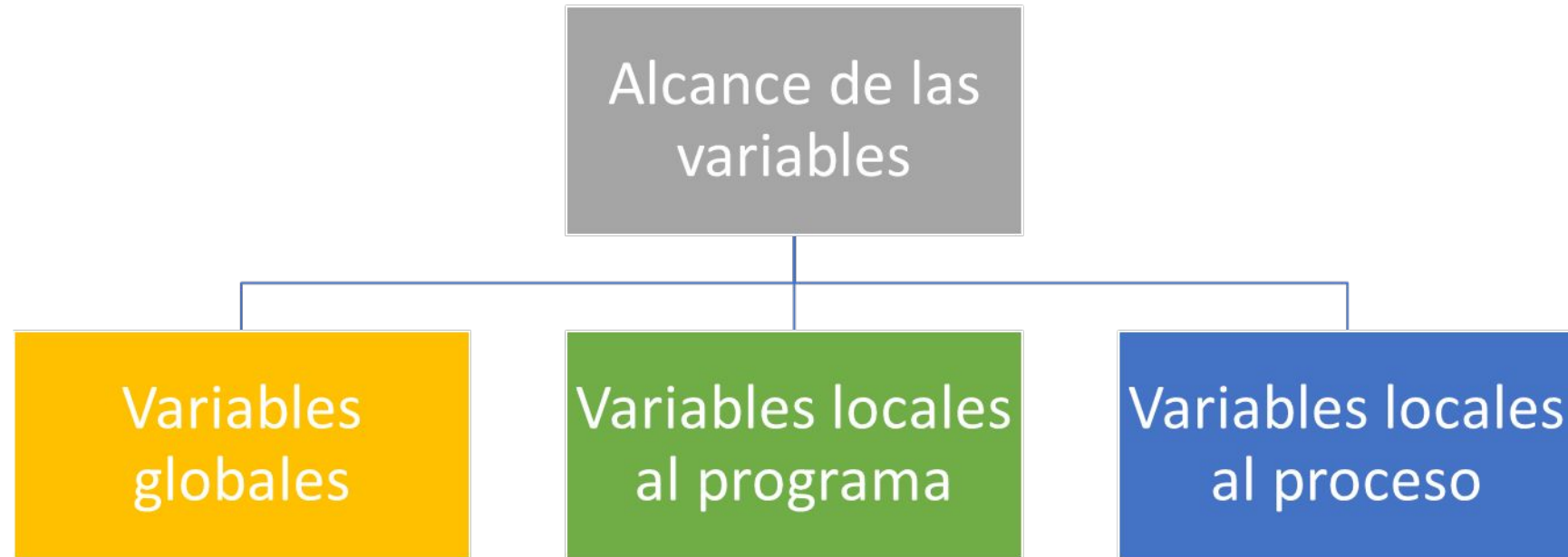
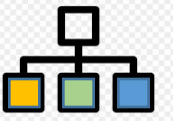
```
end;
```

```
begin
```

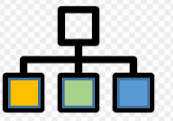
```
  write ('El resultados es,auxiliar);
```

```
end.
```

CADP – MODULARIZACION ALCANCE DE LAS VARIABLES



CADP – MODULARIZACION ALCANCE DE LAS VARIABLES



```
Program alcance;
```

```
Var
```

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

a,b, son **Variables globales** del programa

Pueden ser usadas en todo el programa (incluyendo módulos)

```
procedure prueba;
```

```
Var
```

```
  c: integer;
```

```
Begin
```

```
End.
```

c, es una **variable local del proceso**

Pueden ser usadas sólo en el proceso que están declaradas

```
Var
```

```
  d:integer;
```

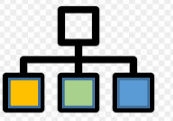
```
Begin
```

```
End.
```

d, es una **variable local del programa**

Pueden ser usadas sólo en el cuerpo del programa

CADP – MODULARIZACION **ALCANCE DE LAS VARIABLES**



```
Program alcance;  
Const  
...  
Type  
...
```

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

```
Procedure prueba;  
Var  
    c: integer;  
Begin  
End.
```

```
Var  
    d:integer;  
Begin  
End.
```

```
Program alcance;  
Const  
...  
Type  
...
```

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

```
Procedure prueba;  
Var  
    c: integer;  
Begin  
End.
```

```
Var  
    d:integer;  
Begin  
End.
```

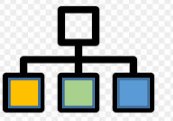
```
Program alcance;  
Const  
...  
Type  
...
```

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

```
Procedure prueba;  
Var  
    c: integer;  
Begin  
End.
```

```
Var  
    d:integer;  
Begin  
End.
```

CADP – MODULARIZACION ALCANCE DE LAS VARIABLES



Qué
imprime?

```
Program alcance;  
Var  
  x,y: integer;  
  
Procedure prueba;  
Var  
  x:integer;  
Begin  
  x:= 34 DIV 3;  
  write (x);  
End;  
Var  
  x:integer;  
Begin  
  x:= 8; y:=9;  
  prueba;  
  write (x);  
  write (y);  
End.
```

Variables de programa (globales)

```
x:=  
y:= 9
```

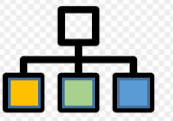
Variables del proceso prueba

```
x:= 11  
Imprime 11
```

Variables del programa (locales)

```
x:= 8  
Imprime 8  
Imprime 9
```

CADP – MODULARIZACION ALCANCE DE LAS VARIABLES



Qué
imprime?

```
Program alcance;  
Var  
  x,y: integer;  
  
Procedure prueba;  
Var  
  x:integer;  
Begin  
  x:= 34 DIV 3;  
  write (x);  
End;  
Var  
  x:integer;  
Begin  
  x:= 8;  
  prueba;  
  write (x);  
  write (y);  
End.
```

Variables de programa (globales)

```
x:=  
y:=
```

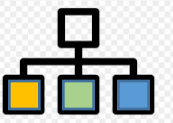
Variables del proceso prueba

```
x:= 11           Imprime 11
```

Variables del programa (locales)

```
x:= 8           Imprime 8  
                Imprime basura
```

CADP – MODULARIZACION ALCANCE DE LAS VARIABLES



Qué
imprime?

```
Program alcance;
```

```
Var
```

```
  x: integer;
```

```
Procedure prueba;
```

```
Var
```

```
  x:integer;
```

```
Begin
```

```
  x:= 34 DIV 3;
```

```
  write (x);
```

```
End;
```

```
Var
```

```
  x:integer;
```

```
Begin
```

```
  x:= 8;
```

```
  prueba;
```

```
  write (x);
```

```
  write (y);
```

```
End.
```

Variables de programa (globales)

```
x:=
```

Variables del proceso prueba

```
x:= 11
```

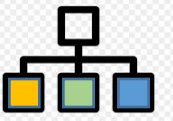
Imprime 11

Variables del programa (locales)

```
x:= 8
```

Imprime 8

Da error



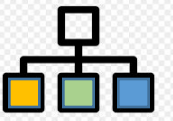
Si es una variable utilizada en un proceso

- Se busca si es variable local
- Se busca si es un parámetro
- Se busca si es variable global al programa

Si es una variable usada en un programa

- Se busca si es variable local al programa
- Se busca si es variable global al programa

CADP – MODULARIZACION **ALCANCE DE LAS VARIABLES**



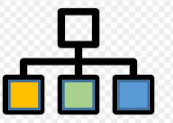
Qué
imprimen?

```
Program uno;  
Var  
  x,a,b: integer;  
procedure prueba;  
  var  
    x: integer;  
  begin  
    x:= 5;  
    write (x);  
  end;  
Begin  
  x:=10;  
  prueba;  
  write (x);  
End.
```

```
Program dos;  
Var  
  x,a,b: integer;  
  
procedure prueba;  
  Begin  
    write (x);  
  End;  
  
Begin  
  x:=5;  
  prueba;  
  write (x);  
End.
```

```
Program tres;  
Var  
  x : char;  
  
procedure prueba;  
  Var  
    x:integer;  
  Begin  
    x:= 4;  
    write (x);  
  End;  
Begin  
  x:='a';  
  prueba;  
  write (x);  
End.
```

CADP – MODULARIZACION ALCANCE DE LAS VARIABLES



```
Program uno;  
Var  
  x,a,b: integer;  
procedure prueba;  
  type  
    días = 1..7;  
  var  
    x: integer;  
  begin  
    x:= 5;  
  end;  
Begin  
  x:=10;  
  prueba;  
  write (x);  
End.
```

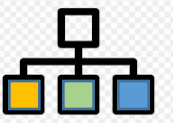


Se puede declarar un tipo nuevo dentro de un módulo?

Si se puede donde puedo declarar variables de ese tipo nuevo?

CADP – MODULARIZACION

ALCANCE DE LAS VARIABLES



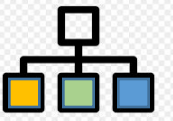
```
Program uno;  
  procedure prueba;  
    procedure auxiliar  
    var  
      ...  
    begin  
    end;  
    var  
      x: integer;  
    begin  
      x:= 5;  
    end;  
  Begin  
    prueba;  
  End.
```

Se puede declarar un procedimiento dentro de otro?

Si se puede, desde donde se puede invocar a ese nuevo procedimiento?

CADP – MODULARIZACION

ALCANCE DE LAS VARIABLES

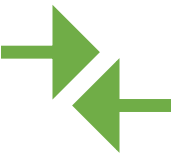


```
Program uno;  
Var  
  x:integer;  
procedure prueba;  
  procedure auxiliar;  
  var  
    ...  
  begin  
    x:= 4;  
  end;  
var  
  x: integer;  
begin  
  x:= 5;  
end;
```

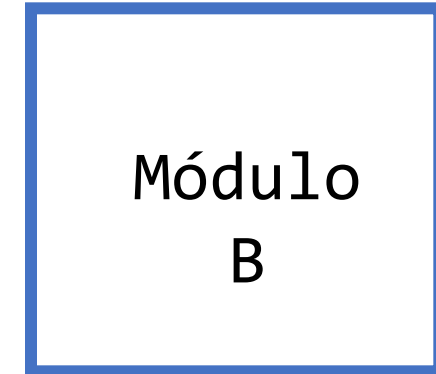
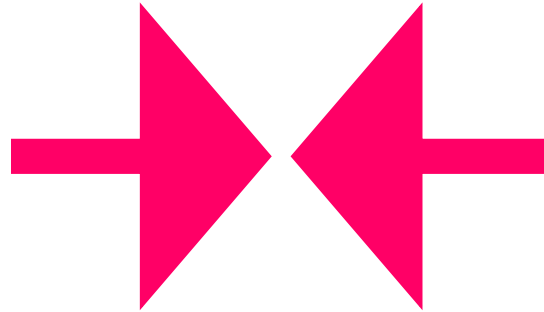
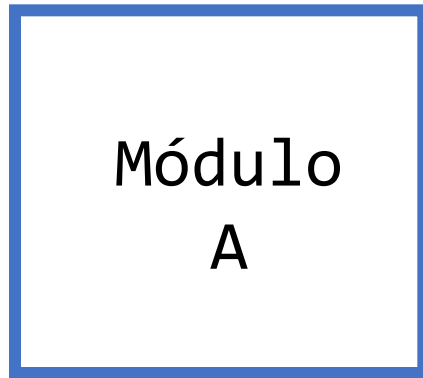
Se puede? A que x se hace referencia ?

Se puede? A que x se hace referencia?

```
Begin  
  prueba;  
End.
```



COMUNICACIÓN ENTRE MODULOS



Variables Globales
Parámetros

*Cuál
utilizamos?*