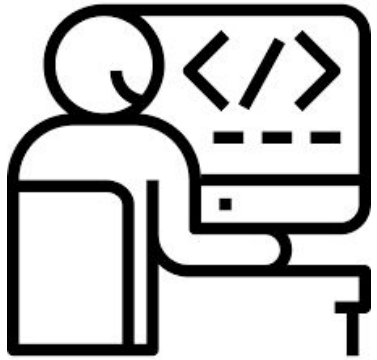


Algoritmos y Programación I

AyPI – **Temas de la clase pasada**

- Etapas para la resolución de problemas
- Algoritmos, precondiciones y postcondiciones
- Datos (constantes y variables)
- Tipos de datos simples definidos por el lenguaje (integer, real, char, boolean)
- Estructuras de control: secuencia y decisión

AyPI – **Ejercicios de repaso**

1. Implementar un programa que lea un número integer e informe si el número ingresado es mayor, menor o igual a 1000.

```
Program repaso1;
const valor = 1000;
var num: integer;

begin
   read(num);
   if (num > valor)
   then write (‘Numero mayor a 1000’)
   else if (num < 1000)
        then write (‘Numero menor a 1000’)
        else write (‘Numero igual a 1000’);
end.
```

AyPI – **Ejercicios de repaso**

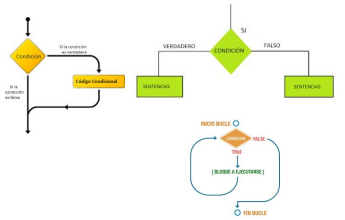
2. Implementar un programa que lea un caracter e informe si el caracter ingresado corresponde a una letra minúscula, una letra mayúscula, un dígito o es cualquier otro símbolo.

```
Program repaso2;
var car: char;

begin
   read(car);
   if (car >= ‘a’) and (car <= ‘z’)
   then write (‘Caracter corresponde a una letra minuscula’)
   else if (car >= ‘A’) and (car <= ‘Z’)
        then write (‘Caracter corresponde a una letra mayuscula)
        else if (car >= ‘0’) and (car <= ‘9’)
             then write (‘Caracter corresponde a un digito’)
             else write (‘Caracter corresponde a un símbolo’);
end.
```

AyPI – **Temas de la clase de hoy**

- Estructura de control FOR
- Estructuras de control WHILE
- Cálculo de máximos y mínimos
- Modularización

AyPI – **ESTRUCTURAS DE CONTROL**



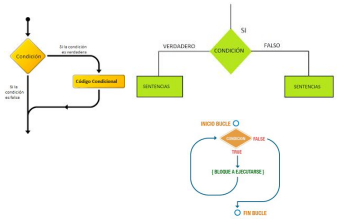
ITERACIÓN

Puede ocurrir que se desee ejecutar un bloque de instrucciones desconociendo el número exacto de veces que se ejecutan.

Para estos casos existen en la mayoría de los lenguajes de programación estructurada las estructuras de control iterativas condicionales.

Como su nombre lo indica las acciones se ejecutan dependiendo de la evaluación de la condición.

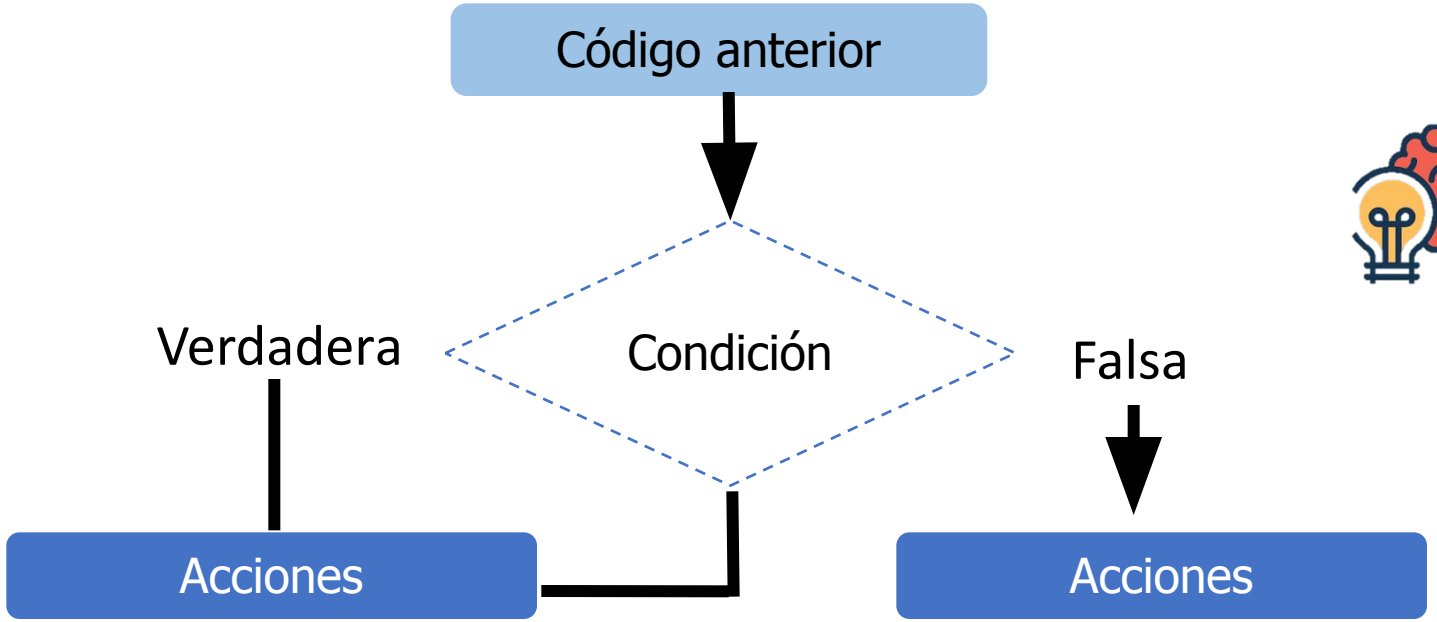
AyPI – **ESTRUCTURAS DE CONTROL**



ITERACIÓN - PRECONDICIONAL

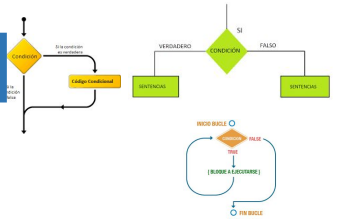
Evalúan la condición y si es verdadera se ejecuta el bloque de acciones. Dicho bloque se pueda ejecutar 0, 1 ó más veces.

Importante: el valor inicial de la condición debe ser conocido o evaluable antes de la evaluación de la condición.



¿A qué estructura de control se parece?.

¿Cómo es la sintaxis?



ITERACIÓN PRECONDICIONAL

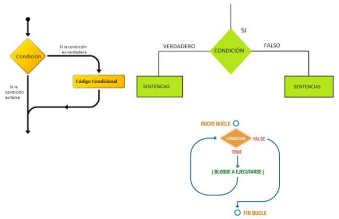
-

```
while (condición) do
   accion;
```

más de una acción

```
while(condición) do
 begin
  acción 1;
  acción 2;
 end;
```

AyPI – **ESTRUCTURAS DE CONTROL** ITERACIÓN



Realizar un programa que lea edades de personas hasta leer una edad igual a 50. Al finalizar informe la suma de las edades pares

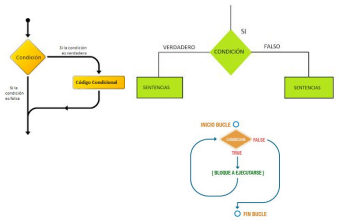
- Cómo leo una edad
- Cómo veo si es par
- Cuál es la condición de fin
- Cómo muestro el resultado

35
70
32
5
50

informa

102

AyPI – ESTRUCTURAS DE CONTROL ITERACIÓN



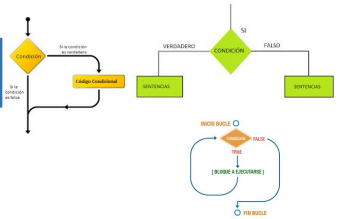
```
Program uno;
var
  resto,edad:integer;
  total:integer;


begin
   total:=0;
   while (edad <> 50)do
    begin
      read(edad);
      resto:= edad MOD 2;
      if (resto = 0)then
        total:= total + edad;
    end;
   write (total);
end.
```

¿Cuál es el error?

```
Program dos;
var
  edad,resto:integer;
  total:integer;

begin
   total:=0;
   read (edad);
   while (edad <> 50)do
    begin
      resto:= edad MOD 2;
      if (resto = 0)then
        total:= total + edad;
      read (edad);
    end;
   write (total);
end.
```

¿Otra alternativa?

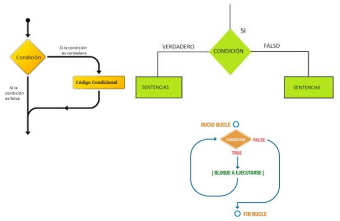
AyPI – ESTRUCTURAS DE CONTROL ITERACIÓN



```
Program tres;
var
  edad:integer;
  total:integer;
begin
   total:=0;
   read (edad);
   while (edad <> 50)do
    begin
     if (edad MOD 2 = 0)then
       total:= total + edad;
     read (edad);
    end;
   write (total);
end.
```

No se utiliza la variable resto

AyPI – **ESTRUCTURAS DE CONTROL** ITERACIÓN



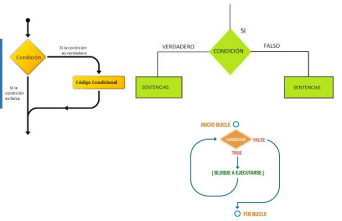
Realizar un programa que lea precios de productos hasta leer un precio igual a 0. Al finalizar informar el promedio de los precios leídos.

- Cómo leo un precio
- Cómo calculo el promedio
- Cuál es la condición de fin
- Cómo muestro el resultado

```
100,50

250.00

85.25

0
```

informa **142.25**

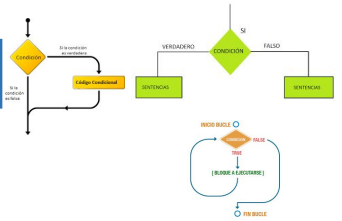
AyPI – ESTRUCTURAS DE CONTROL ITERACIÓN



```
Program promedioPrecios;
var precio, prom, suma: real;
    cant: integer;
begin
   cant:=0; suma:= 0;
   read (precio);
   while (precio <> 0) do begin
     suma:= suma + precio;
     cant:= cant + 1;
     read (precio);
   end;
   if (cant = 0)
   then write ('No se leyeron precios')
   else begin
          prom:= suma/cant;
          write ('Promedio de precios: ', prom);
        end;
end.
```

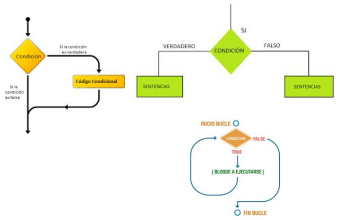
¿Otra alternativa?

AyPI – ESTRUCTURAS DE CONTROL ITERACIÓN



```
Program promedioPreciosOtraAlternativa;
var precio, suma: real;
    cant: integer;
begin
   cant:=0;
   read (precio); suma:= 0;
   while (precio <> 0) do begin
     suma:= suma + precio;
     cant:= cant + 1;
     read (precio);
   end;
   if (cant = 0)
   then write ('No se leyeron precios')
   else write ('Promedio de precios: ', suma/cant);
end.
```

AyPI – **ESTRUCTURAS DE CONTROL**



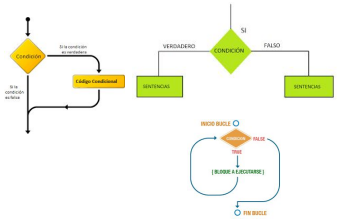
Dados los siguientes enunciados, ¿qué estructuras de control usaría?

- Realizar un programa que lea un número e informe si el número es un múltiplo de 3.
- Realizar un programa que lea caracteres hasta leer el caracter “@” e informe la cantidad de letras ‘a’ leídas.

¿Cómo lo hago?

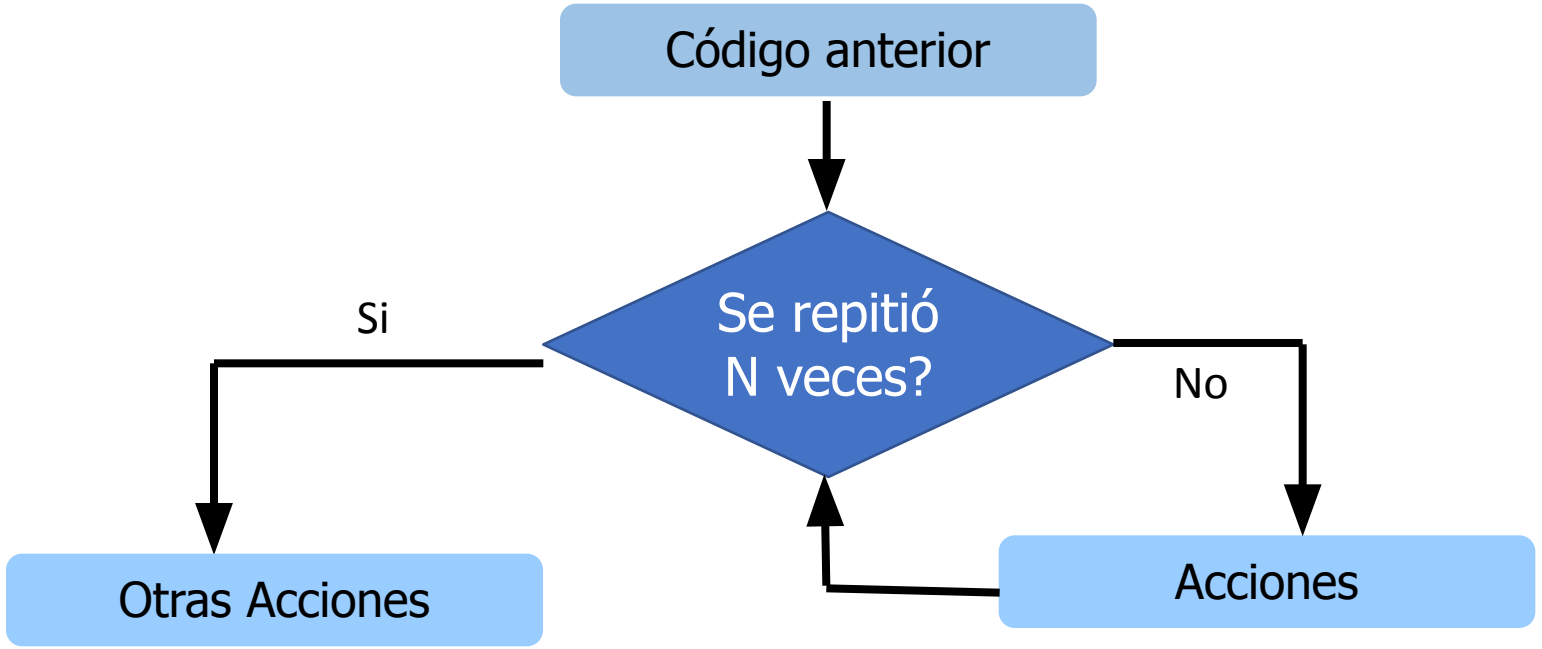
Realizar un programa que lea 10 números e informe la suma.

AyPI – **ESTRUCTURAS DE CONTROL**



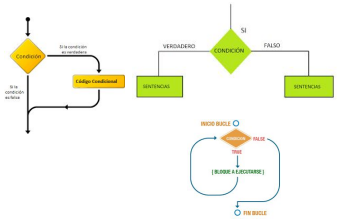
REPETICIÓN

Es una extensión natural de la secuencia. Consiste en repetir N veces un bloque de acciones.
Este número de veces que se deben ejecutar las acciones es fijo y conocido de antemano



¿Cómo es la sintaxis?

AyPI – **ESTRUCTURAS DE CONTROL REPETICIÓN**



```
for variable_indice :=  valor_inicial to valor_final do
     accion 1;
```

más de una acción

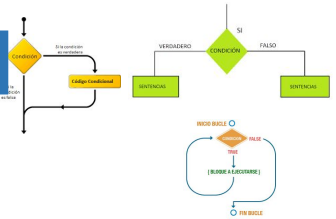
```
for variable_indice :=  valor_inicial to valor_final do
  begin
     accion 1;
     accion 2;
  end;
```

¿Qué es la variable_índice?

¿Dónde se declara?

¿Qué son valor_inicial y valor_final?

AyPI – **ESTRUCTURAS DE CONTROL** REPETICIÓN



```
Ejemplo 1:
For i := 1 to  10 do
        acción;
```

¿De qué tipo es el índice i?
¿qué valores toma i?

```
Ejemplo 2:
For i := ‘A’  to  ‘H’ do
        acción;
```

¿De qué tipo es el índice i?
¿qué valores toma i?

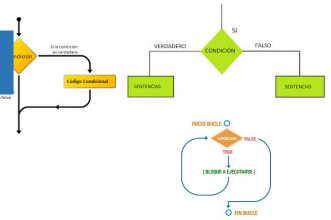
```
Ejemplo 3:
For i:= False  to  True do
        acción;
```

¿De qué tipo es el índice i?
¿qué valores toma i?

```
Ejemplo 4:
For i := 20 to 18 do
    acción;
```

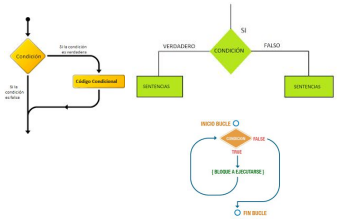
```
Ejemplo 5:
For índice := 20 downto 18 do
        begin
           acción;
           acción;
        end;
```

AyPI – **ESTRUCTURAS DE CONTROL** **REPETICIÓN**



- La variable índice debe ser de tipo ordinal
- La variable índice no puede modificarse dentro del lazo
- La variable índice se incrementa y decrementa automáticamente
- Cuando el for termina la variable índice no tiene valor definido.

AyPI – ESTRUCTURAS DE CONTROL REPETICIÓN



Realizar un programa que lea precios de 10 productos que vende un almacén. Al finalizar el programa debe informar la suma de todos los precios leídos.

- ¿Qué tipo de datos utilizaría para representar el precio?
- ¿Cuál es la condición de fin?
- ¿Cómo calculo la suma?

```
100,5
56,5
15
10
12,5
14
7,5
150,00
25,40
78,50
```

informa **474,4**

AyPI – **ESTRUCTURAS DE CONTROL** REPETICIÓN

```
Program uno;
var
   precio,total:real;
   i:integer;
begin
   total := 0;
   for i:= 1 to 10 do
     begin
         read (precio);
         total:= total + precio;
     end;
   write ('La suma de los precios de los
           productos del almacén es: ',total);
end.
```

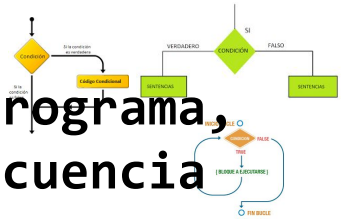
¿Qué modificaría si quiere informar al final , también el precio del 5to producto leído?

AyPI – **ESTRUCTURAS DE CONTROL** REPETICIÓN

```
Program uno;
var
   quinto,precio,total:real;
   i:integer;
begin
   total := 0;
   for i:= 1 to 10 do
     begin
         read (precio);
         if (i=5) then
             quinto:= precio;
         total:= total + precio;
     end;
   write ('La suma de los precios de los
           productos del almacén son: ',total);
   write ('El precio del quinto producto es: ',quinto);
end.
```

AyPI – **ESTRUCTURAS DE CONTROL**

```
Program uno;
var
   i,num1,num2:integer;
Begin
  num2:= 0;
  for i:= 1 to 5 do
     begin
         read (num1);
         while (num1 mod 2 = 0) do
           begin
             num2:= num2 + 1;
             read (num1);
           end;
     end;
    write (num2);
end.
```



¿Qué imprime el programa, si se lee esta secuencia de números?

4
8
126
5
3
6
1
1568
6
10
7
19
22
24
3

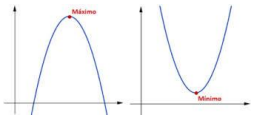
AyPI – **ESTRUCTURAS DE CONTROL**

```
Program uno;
var
   i,j,num1,num2:integer;
Begin
  num2:= 0;
  for i:= 1 to 3 do
     begin
         read (num1);
         for j:= 1 to 2 do
          begin
            if (num1 mod 2 = 1) then
              num2:= num2+1;
            read (num1);
         end;
         read (num1);
     end;
   write (num2);
end.
```

¿Qué imprime el programa, si se lee esta secuencia de números?

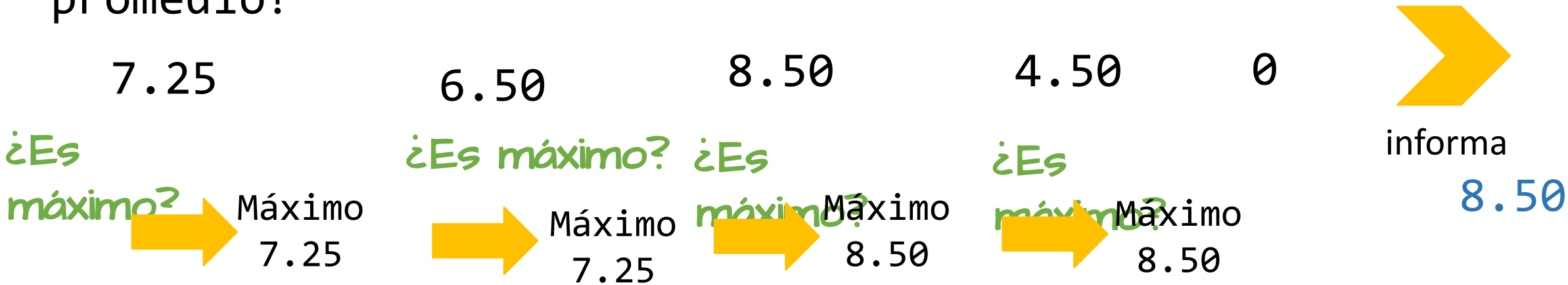
4
7
126
5
3
6
1
1568
6
10
7
19
22
24
3

AyPI – **MAXIMOS y MINIMOS** **MAX-MIN**

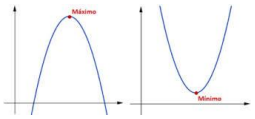


Realizar un programa que lea promedios de notas de alumnos hasta leer un promedio igual a 0. Al finalizar informar el promedio más alto.

- ¿Con qué tipo de dato represento el promedio?
- ¿Cuál es la condición de fin?
- ¿Cómo verifico que el nuevo promedio leído es el mejor promedio?



AyPI – **MAXIMOS y MINIMOS** **MAX-MIN**



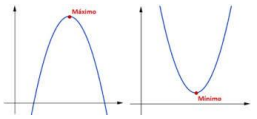
Realizar un programa que lea promedios de notas de alumnos hasta leer un promedio igual a 0. Al finalizar informar el promedio más alto.

```
Program uno;
var
  prom:real;
begin
  Leo un promedio (prom);
  while (no sea la condición de fin) do
    begin
      verificar si es máximo
      Leo un promedio (prom);
    end;

   write ('El mejor promedio es: ',     );
end.
```

¿Cómo verifico si es máximo?

AyPI – MAXIMOS y MINIMOS

MAX-MIN



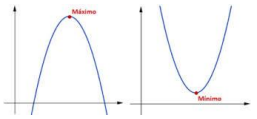
Realizar un programa que lea promedios de notas de alumnos hasta leer un promedio igual a 0. Al finalizar informar el promedio más alto.

```
Program uno;
var
  prom:real;   max:real;
begin
  Leo un promedio (prom);
  while (no sea la condición de fin) do
    begin
      verificar si es máximo    If (prom >= max) then
                                  max:= prom;
      Leo un promedio (prom);
    end;

   write (“El mejor promedio es:”, max );
end.
```

¿Cuál es el error?

AyPI – MAXIMOS y MINIMOS

MAX-MIN

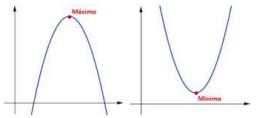


Realizar un programa que lea promedios de notas de alumnos hasta leer un promedio igual a 0. Al finalizar informar el promedio más alto.

```
Program uno;
var
  prom:real;   max:real;
begin
   max:= -1;
  Leo un promedio (prom);
  while (no sea la condición de fin) do
    begin
      verificar si es máximo    If (prom >= max) then
                                  max:= prom;
      Leo un promedio (prom);
    end;

   write (“El mejor promedio es:”, max  );
end.
```

AyPI – MAXIMOS y MINIMOS

MAX-MIN

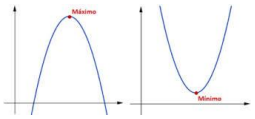


Realizar un programa que lea promedios de notas de alumnos hasta leer un promedio igual a 0. Al finalizar informar el promedio más alto.

```
Program uno;
var
  prom:real;   max:real;
begin
  max:= -1;
  read(prom);
  while (prom <> 0) do
    begin
      If (prom >= max) then
        max:= prom;
      read (prom);
    end;
   write ('El mejor promedio es:',max  );
end.
```

¿Qué modifico si quiero saber el número del alumno con mejor promedio?

AyPI – **MAXIMOS y MINIMOS** **MAX-MIN**



```
Program uno;
var
  prom,max:real; nro_alu:integer; max_nro_alu:integer;
begin
  max:= -1;
  read(prom); read(nro_alu);
  while (prom <> 0) do
    begin
      if (prom >= max) then begin
        max:= prom;
        max_nro_alu:= nro_alu;
      end;
      read(prom); read(nro_alu);
    end;
   write ('El mejor promedio es:',max,'y el alumno es', max_nro_alu);
end.
```

AyPI – **MAXIMOS y MINIMOS** RECORDAR

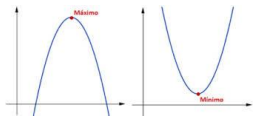
Utilizar una variable que representará al máximo.

Inicializar la variable máximo en un valor bajo antes de comenzar la lectura de los datos.

Actualizar la variable máximo cuando corresponda

AyPI – **MAXIMOS y MINIMOS**

MAX-MIN



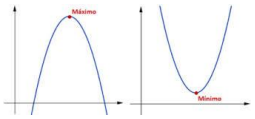
```
Program uno;
var
  num,max:integer;
begin
  max:= 0;
  read(num);
  while (num <> 80) do
    begin
      if (num >= max) then begin
        max:= num;
      end;
      read(num);
    end;
   write ('El número más alto es :',max);
end.
```

¿Qué imprime si se lee?

```
23
8
23
5
0
80
```

AyPI – **MAXIMOS y MINIMOS**

MAX-MIN

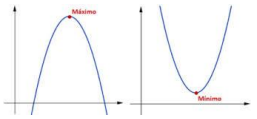


```
Program dos;
var
  cantidad,codigo,max_cant,max_codigo:integer;
begin
  max_cant:= 0;
  read(codigo); read(cantidad);
  while (num <> 80) do
    begin
      if (cantidad > max_cant) then begin
        max_codigo:= codigo;
        max_cant:= cantidad;
      end;
      read(codigo); read(cantidad);
    end;
   write ('El codigo con mas cantidad es: ',max_codigo);
end.
```

¿Qué imprime si se lee?

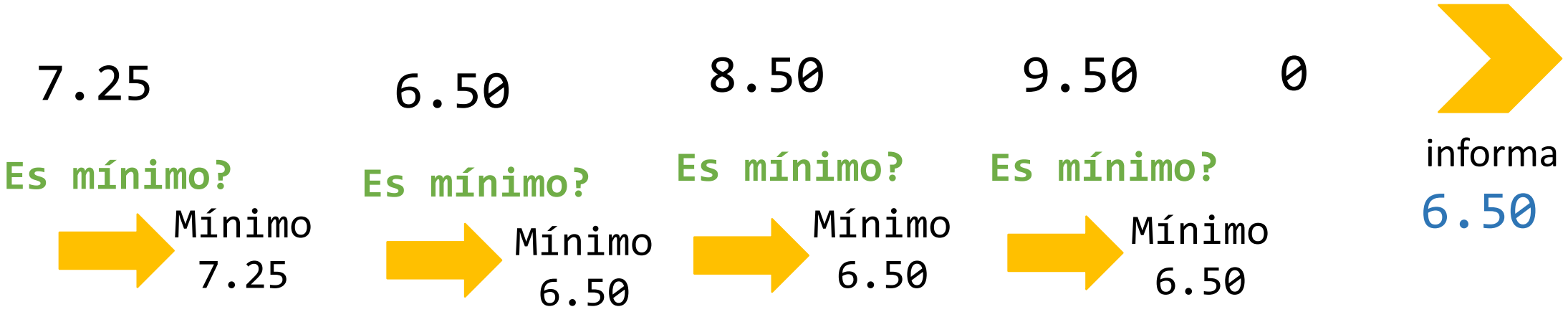
15	0
23	0
8	0
80	16

AyPI – **MAXIMOS y MINIMOS** **MAX-MIN**



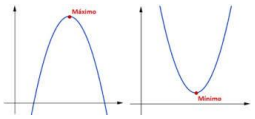
Realizar un programa que lea promedios de notas de alumnos hasta leer un promedio igual a 0. Al finalizar informar el promedio más bajo.

- ¿Qué valor es el promedio?
- ¿Cuál es la condición de fin?
- ¿Cómo verifico que sea el mejor promedio?



AyPI – MAXIMOS y MINIMOS

MAX-MIN



Realizar un programa que lea promedios de notas de alumnos leer un promedio igual a 0. Al finalizar informar el promedio más bajo.

```
Program minimo;
Var
   min:real;
   prom:real;
begin
   min:= 11;
   read(prom);
   while (prom <> 0) do
     begin
       If (prom <= min) then
         min:= prom;
       read (prom);
     end;
    write ('El peor promedio es: ', min );
end.
```

AyPI – **MAXIMOS y MINIMOS** RECORDAR

Utilizar una variable que representará al mínimo.

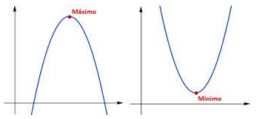
Inicializar la variable mínimo en un valor alto antes de comenzar la lectura de los datos.

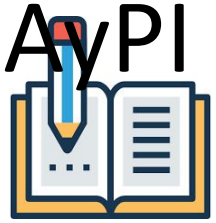
Actualizar la variable mínimo cuando corresponda

¿Qué modifico si quiero saber el promedio máximo y el promedio mínimo?

AyPI – MAXIMOS y MINIMOS

MAX-MIN





Realizar un programa que lea promedios de notas de alumnos hasta leer un promedio igual a 0. Al finalizar informar el promedio más bajo y el más alto.

```
Program minMax1;
Var
  min,max,prom:real;
begin
  min:= 11; max:=-1;
  read(prom);
  while (prom <> 0) do
    begin
      if (prom >= max) then
        max:= prom
      else
       if (prom<= min) then
         min:= prom;
     read (prom);
    end;
   write (min,max);
end.
```

Funcionan las dos?

Si se leen los valores a continuación, qué imprime cada uno?

5.50
10
8.50
0

```
Program minMax2;
Var
  min,max,prom:real;
begin
  min:= 11; max:=-1;
  read(prom);
  while (prom <> 0) do
    begin
      if (prom >= max) then
        max:= prom
      if (prom<= min) then
         min:= prom;
     read (prom);
    end;
   write (min,max);
end.
```

AyPI – **MODULARIZACIÓN**

Los problemas del mundo real implican:

- **Complejidad**
- **Extensión**
- **Modificaciones**

Los tratamos de resolver con:

- **Abstracción**
- **Descomposición**
- **Independencia Funcional**

AyPI – **MODULARIZACIÓN**

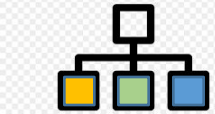
MODULARIZAR

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.

MODULARIZAR



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.

AyPI – **MODULARIZACIÓN**

MÓDULO

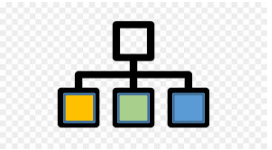
Tarea específica bien definida se comunican entre sí adecuadamente y cooperan para conseguir un objetivo común.

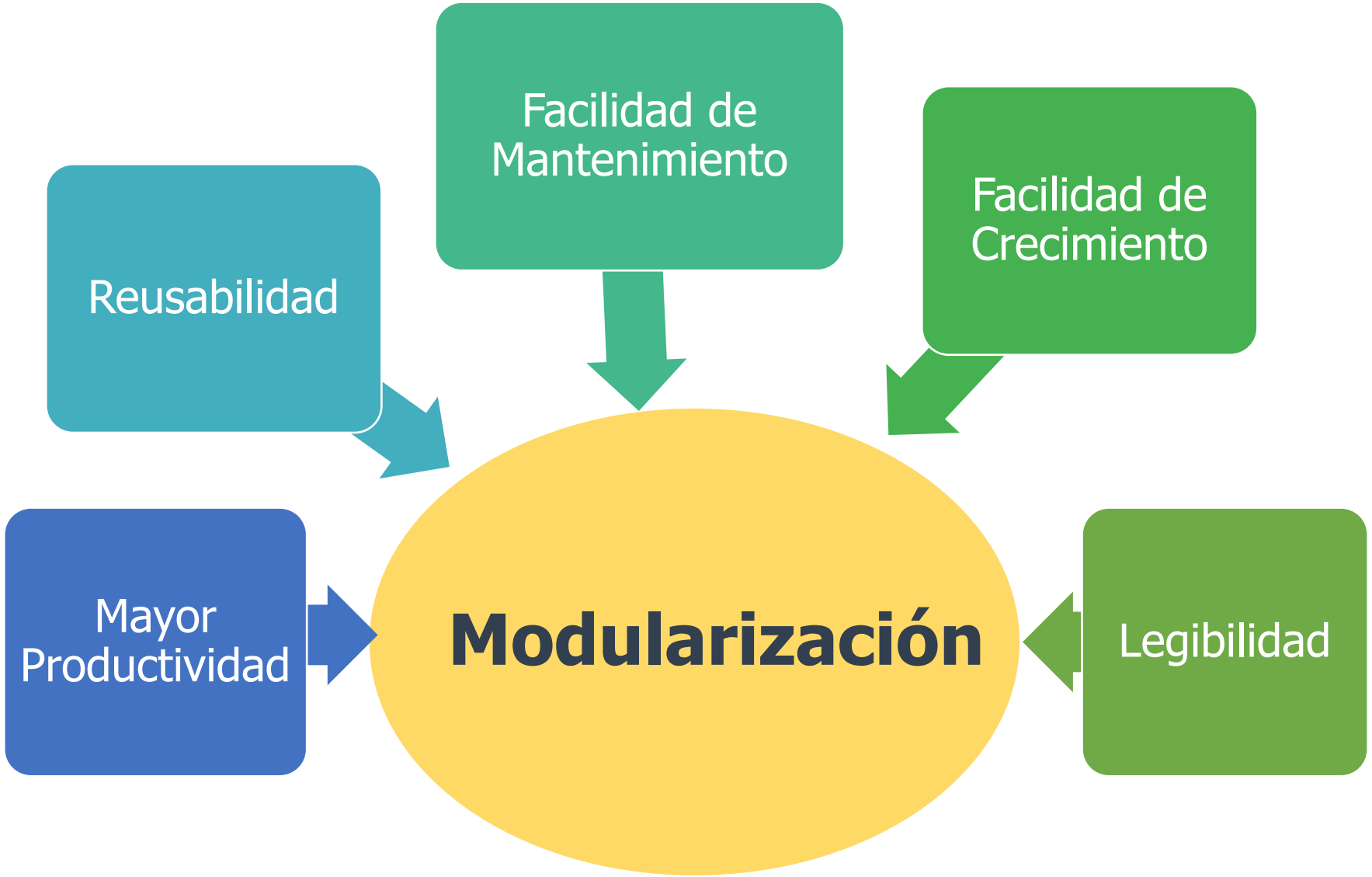
Encapsula acciones tareas o funciones.

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

AyPI – **MODULARIZACIÓN**

VENTAJAS

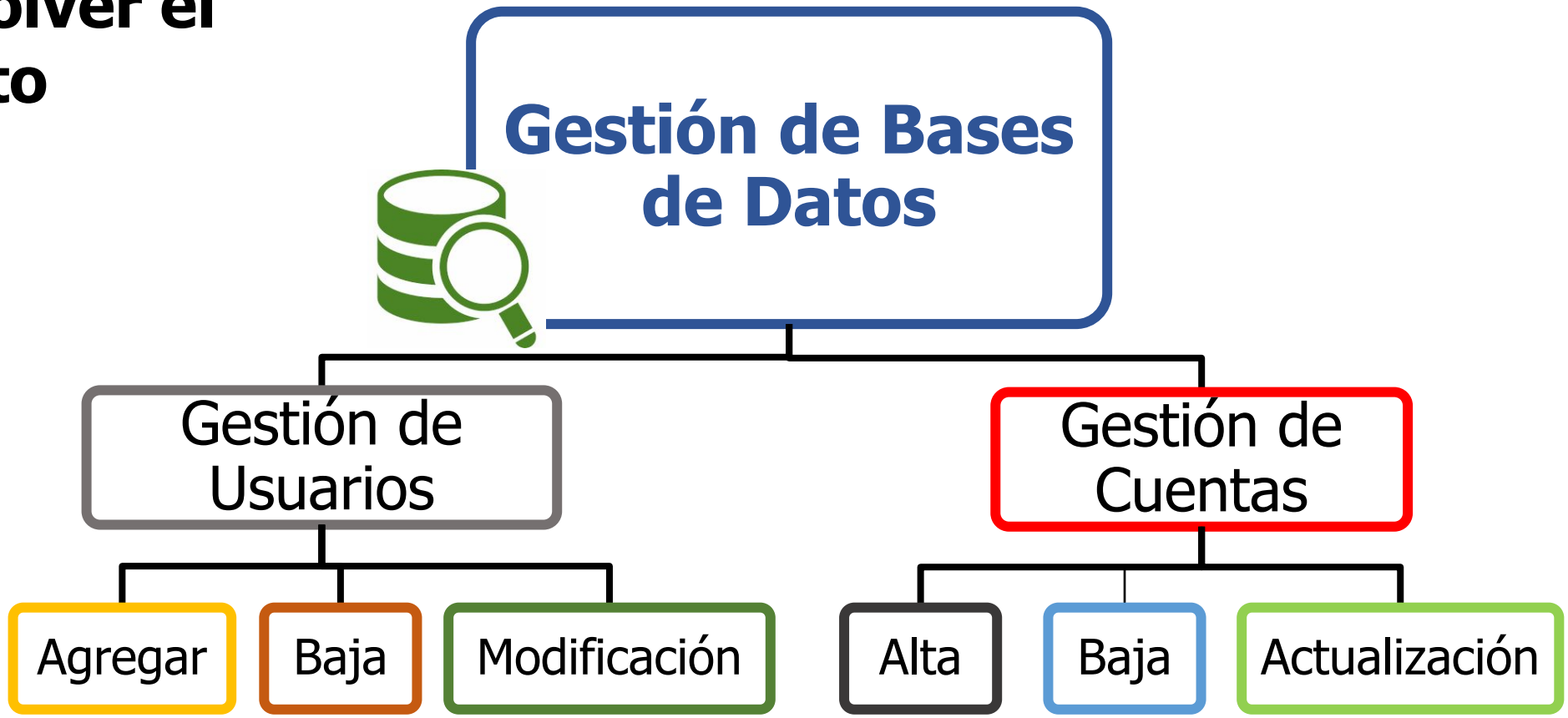




AyPI – **MODULARIZACIÓN**

VENTAJAS

Supongamos que tenemos que resolver el siguiente proyecto

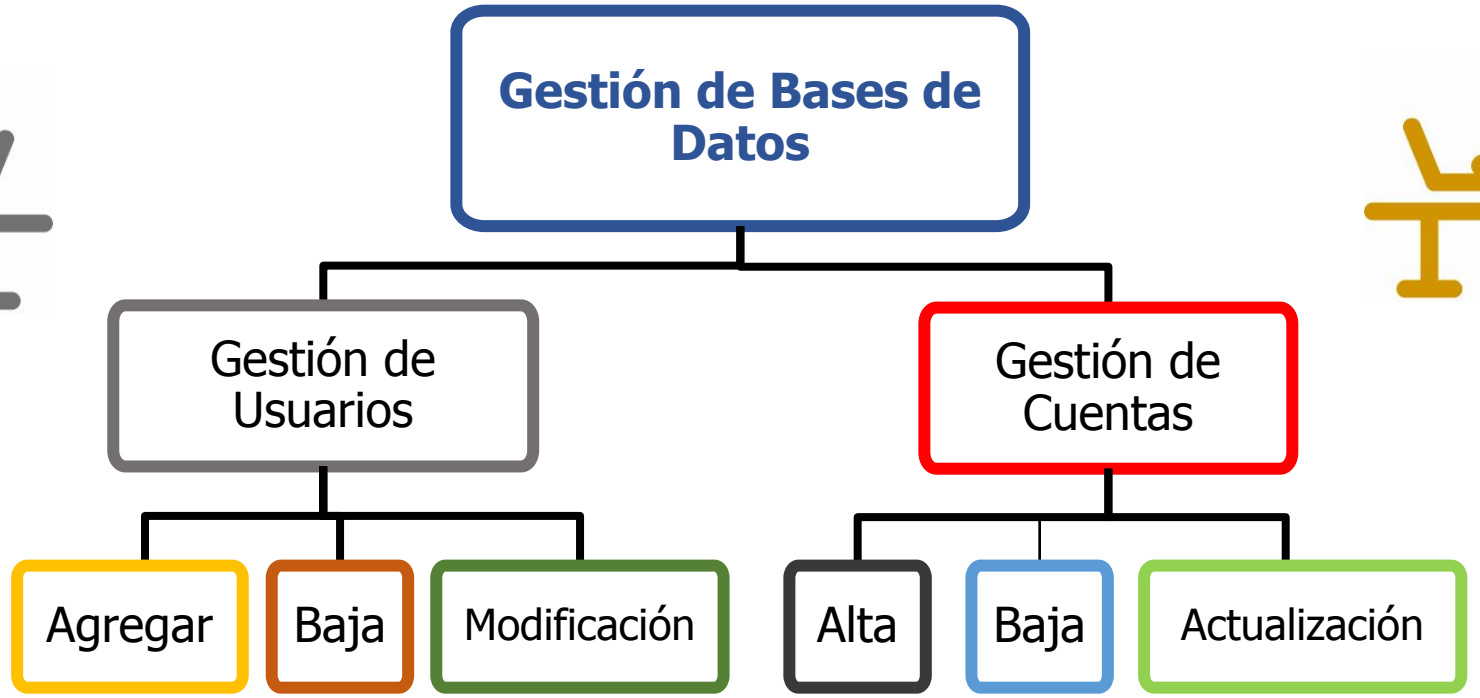


AyPI – **MODULARIZACIÓN**

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

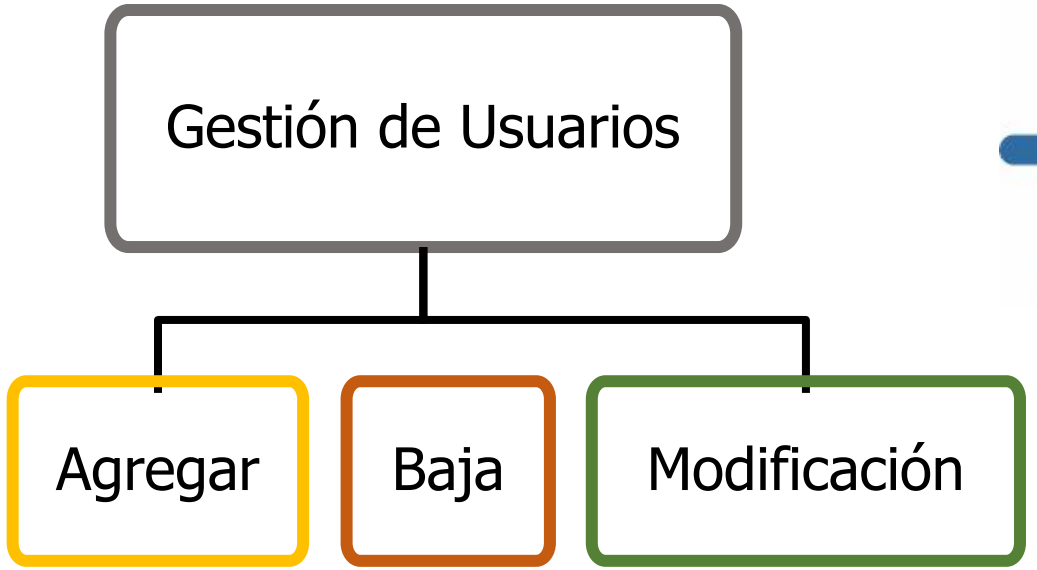


AyPI – **MODULARIZACIÓN**

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

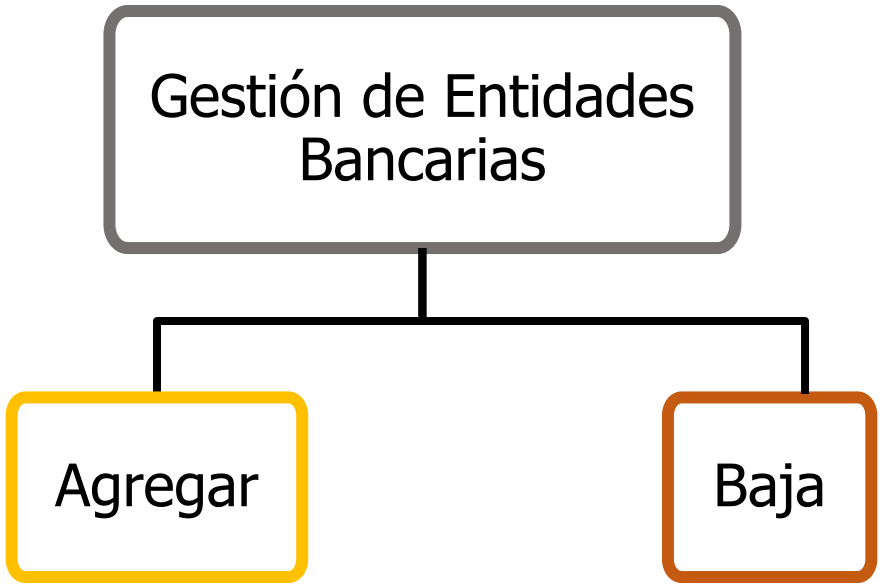


AyPI – **MODULARIZACIÓN**

VENTAJAS

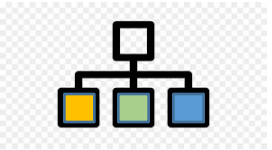
Los sistemas de software reales crecen (es decir aparecen con el tiempo nuevos requerimientos del usuario). La modularización permite disminuir los riesgos y costos de incorporar nuevas prestaciones a un sistema en funcionamiento.

Facilidad de crecimiento



AyPI – **MODULARIZACIÓN**

VENTAJAS

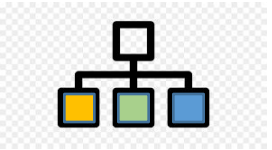


Un efecto de la modularización es una mayor claridad para leer y comprender el código fuente. El ser humano maneja y comprende con mayor facilidad un número limitado de instrucciones directamente relacionadas.

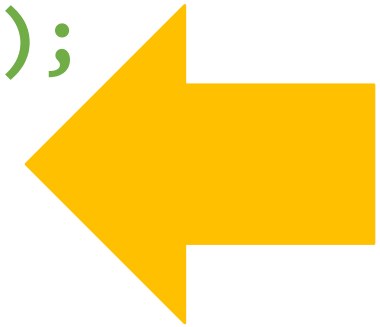
Legibilidad

AyPI – **MODULARIZACIÓN**

PROCEDIMIENTOS



```
Program nombre;
type
   ...
   procedure nombre();
    var ...
    begin
    end;
var ...
begin
end.
```



¿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;
```

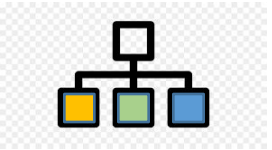
Variables locales

Código del procedimiento

¿Cómo se invocan?

AyPI – **MODULARIZACIÓN**

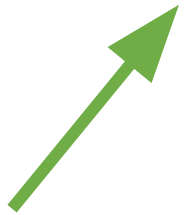
PROCEDIMIENTOS

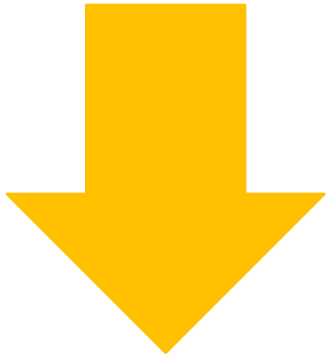


INVOCACIÓN

```
Program uno;
Const
  ....
Type
  ....
procedure auxiliar();
Var ...

begin
  ...
end;

Var
  ....
Begin
   auxiliar(...);
End.
```

Por su nombre





Existe otra forma de modularizar FUNCIONES

FUNCIONES

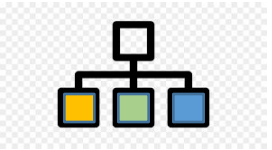
FUNCIÓN

Conjunto de instrucciones que realizan una tarea específica y retorna un único valor.

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

¿Qué diferencias se ven con los procedimientos?

FUNCIONES



INVOCACIÓN POR SU NOMBRE

Invocación usando variable

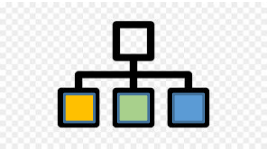
El resultado se asigna a una variable del mismo tipo que devuelve la función.

```
program uno;
Function auxiliar(...): real;
Var ...

begin
    ...
    auxiliar:= ...;
end;
Var
   aux:real;
begin
    aux:= auxiliar(...);
    write (aux);
end.
```

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

AyPI – **MODULARIZACIÓN**

FUNCIONES



INVOCACIÓN 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 …

begin
    ...
    auxiliar:= ...;
end;
Var
  aux:real;
begin
    while (auxiliar(...) = 5.5) do
          begin
          end;
    if (auxiliar(...) = 5.5) then
          begin
          end;
end.
```

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