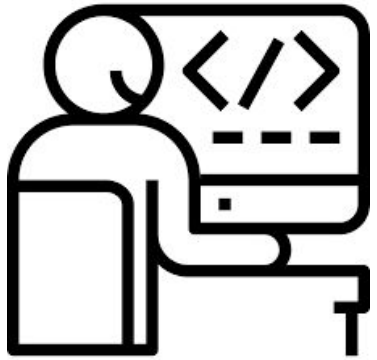


Algoritmos y Programación I

AyPI – **Temas de la clase pasada**

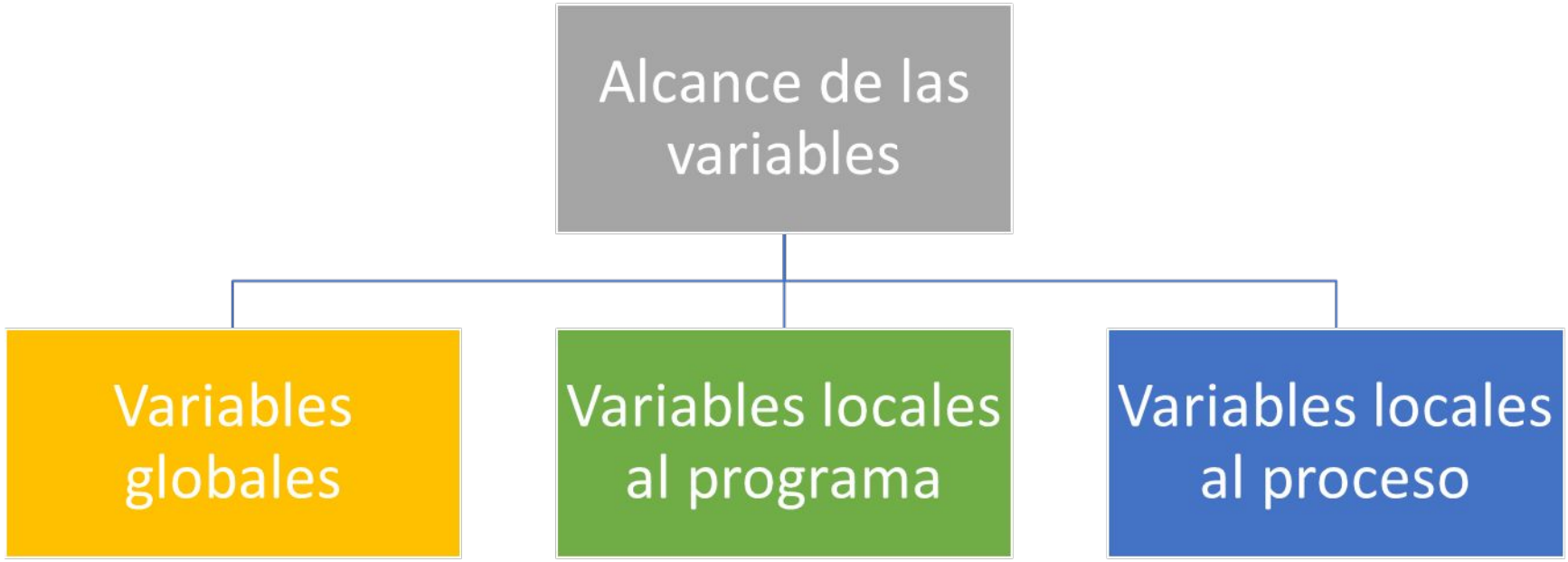
- Estructura de control FOR
- Estructura de control WHILE
- Cálculo de máximos y mínimos
- Modularización

AyPI – **Temas de la clase de hoy**

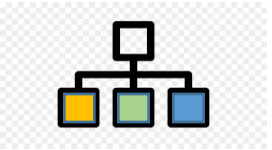
- Alcance de variables
- Comunicación entre módulos

AyPI – **MODULARIZACIÓN**

ALCANCE DE LAS VARIABLES



AyPI – **MODULARIZACIÓN** ALCANCE DE LAS VARIABLES



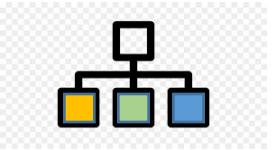
```
Program alcance;

Var
   a,b: integer;

procedure prueba;
Var
  c: integer;
Begin
End.

Var
  d:integer;
Begin
End.
```

a,b, son **Variables globales** del programa → Pueden ser usadas en todo el programa (incluyendo módulos)

c, es una **variable local del proceso** → Pueden ser usadas sólo en el proceso que están declaradas

d, es una **variable local del programa** → Pueden ser usadas sólo en el cuerpo del programa

AyPI – **MODULARIZACIÓN** ALCANCE DE LAS VARIABLES



```
Program alcance;
Const
…
var
    a,b: integer;

Procedure prueba;
Var
  c: integer;
Begin
End.

Var
  d:integer;
Begin
End.
```

```
Program alcance;
Const
    …
Var
    a,b: integer;

Procedure prueba;
Var
  c: integer;
Begin
End.

Var
  d:integer;
Begin
End.
```

```
Program alcance;
Const
…

Var
    a,b: integer;

Procedure prueba;
Var
  c: integer;
Begin
End.

Var
  d:integer;
Begin
End.
```

AyPI – **MODULARIZACIÓN**

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

AyPI – **MODULARIZACIÓN**

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

AyPI – **MODULARIZACIÓN**

ALCANCE DE LAS VARIABLES

Para pensar...

¿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:=

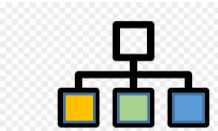
Variables del proceso prueba

x:=

Variables del programa (locales)

x:= Da error de compilación, la variable ‘y’ no está declarada

AyPI – **MODULARIZACIÓN** ALCANCE DE LAS VARIABLES



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

AyPI – **MODULARIZACIÓN** ALCANCE DE LAS VARIABLES

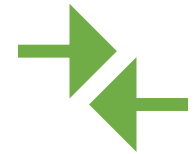
Para pensar...

¿Qué imprimen?

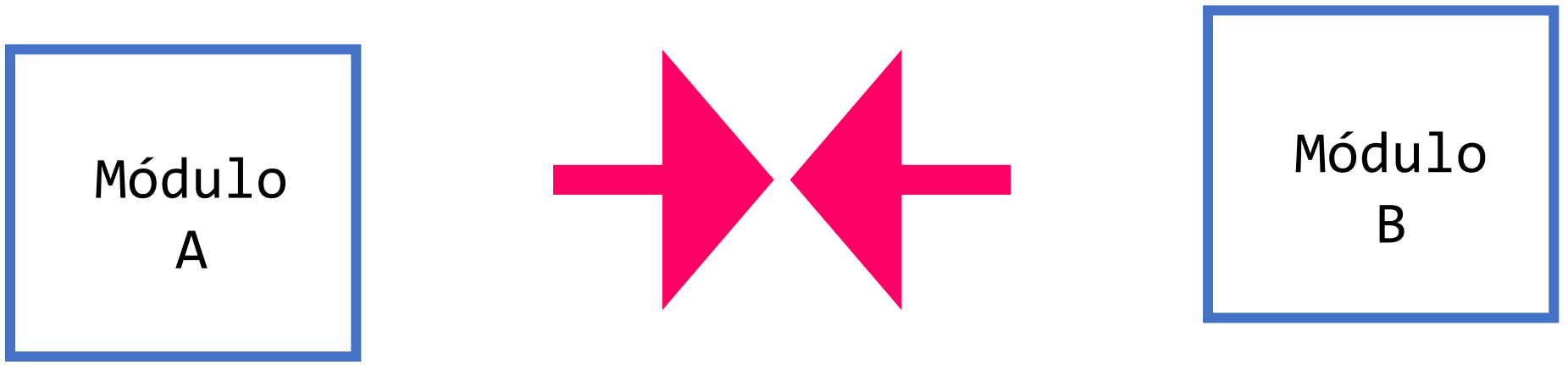
```
Program uno;
Var
 x,a,b: integer;
procedure prueba;
   var
     x: integer;
   begin
      x:= 5;
      write (x);
   end;
Begin
  x:=10;
  prueba;
  write (x);
End.
```

```
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.
```



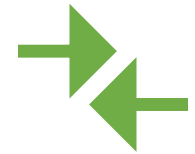
COMUNICACIÓN ENTRE MÓDULOS



Variables Globales

Parámetros

¿Cuál utilizamos?



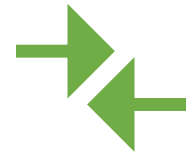
VARIABLES GLOBALES

Demasiados identificadores

No se especifica la comunicación entre los módulos

Conflictos de nombres de identificadores utilizados por diferentes programadores.

Posibilidad de perder integridad de los datos, al modificar involuntariamente en un módulo datos de alguna variable que luego deberá utilizar otro módulo.



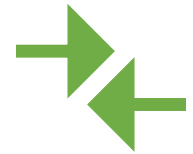
PARÁMETROS

La solución a estos problemas ocasionados por el uso de variables globales es una combinación de ocultamiento de datos (Data Hiding) y uso de parámetros.

El ocultamiento de datos significa que los datos exclusivos de un módulo NO deben ser "visibles" o utilizables por los demás módulos.

El uso de parámetros significa que los datos compartidos se deben especificar como parámetros que se trasmiten entre módulos.

AyPI – **MODULARIZACIÓN** COMUNICACIÓN



PARÁMETROS – ¿Cómo vamos a trabajar?

1. Se analiza para cada módulo entonces: ¿cuáles son los datos propios? y ¿cuáles son los datos compartidos?
2. Los datos propios se declararan locales al módulo.
3. Los datos compartidos se declararán como parámetros.
   - Parámetros por valor
   - Parámetros por referencia

PARÁMETRO POR VALOR

Un dato de entrada por valor es llamado parámetro IN y significa que el módulo recibe (sobre una variable local) un valor proveniente de otro módulo (o del programa principal).

Con él puede realizar operaciones y/o cálculos, pero no producirá ningún cambio ni tampoco tendrá incidencia fuera del módulo.

¿Cómo se declaran?

¿Cómo se usan?

PARÁMETRO POR VALOR

```
procedure uno (nombre1: tipo; nombre2: tipo);

  var
     …

  Begin

    Uso de los parámetros con nombre1 y nombre2

  End;
```

AyPI – **MODULARIZACIÓN**

COMUNICACIÓN

PARÁMETRO POR VALOR

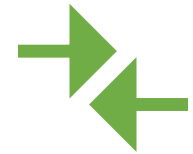
```
Program porValor;

procedure uno (num: integer);
Begin
     if (num = 7)  then
       num:= num + 1;
     write (num);
end;
var
   x: integer;
begin
  x:= 7;
  uno (x);
end.
```

Dentro del procedimiento uno, el parámetro num copia el valor enviado por x (variable del programa)

¿Cómo funciona?

AyPI – **MODULARIZACIÓN**

COMUNICACIÓN



```
Program porValor;

procedure uno (num: integer);
Begin
     if (num = 7)  then
        num:= num + 1;
     write (num);
end;
var
    x: integer;
begin
   x:= 7;
   uno (x);
   write(x);
end.
```

¿Qué pasa si después de llamar al procedimiento uno en el programa imprimo num?

Procedimiento uno
Variables locales
Parámetros

num = ~~8~~

Imprime 8

Programa ppal
Variables globales
Variables de prog

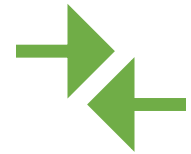
x = 7

Imprime 7

MEMORIA

AyPI – **MODULARIZACIÓN**

COMUNICACIÓN



```
Program porValor;

procedure uno (num: integer);
Begin
     if (num = 7)  then
        num:= num + 1;
     write (num);
end;
var
    num: integer;
begin
   num:= 7;
   uno (num);
   write(num);
end.
```

¿Qué pasa si después de llamar al procedimiento uno en el programa imprimo num?

Procedimiento uno
Variables locales
Parámetros

num = ~~8~~

Imprime 8

Programa ppal
Variables globales
Variables de prog

num = 7

Imprime 7

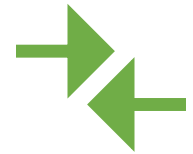
MEMORIA

AyPI – **MODULARIZACIÓN**

```
Program porValor;

procedure uno (x: integer);
Begin
    if (x = 7)  then
      x:= x + 1;
    write (x);
end;
var
   x: integer;
begin
  x:= 7;
  uno (x);
  write(x);
end.
```

¿Qué valores imprimen?

COMUNICACIÓN



```
Program porValor;
procedure uno (num: integer);
Var
 x:integer;
Begin
    if (num = 7) then
       num:= num + 1;
    x:= num;
    write (num); write (x);
end;
var
   x: integer;
begin
  x:= 7;
  uno (x);
  write(x);
end.
```

PARÁMETRO POR REFERENCIA

La comunicación por referencia (OUT, INOUT) significa que el módulo recibe el nombre de una variable (referencia a una dirección) conocida en otros módulos del sistema.

Puede operar con ella y su valor original dentro del módulo, y las modificaciones que se produzcan se reflejan en los demás módulos que conocen la variable.

¿Cómo se declaran?

¿Cómo se usan?

PARÁMETRO POR REFERENCIA

```
procedure uno (var nombre1: tipo; var nombre2: tipo);

   var
      …

   Begin

     Uso de los parámetros con nombre1 y nombre2

   End;
```

AyPI – **MODULARIZACIÓN**

COMUNICACIÓN

PARÁMETRO POR REFERENCIA

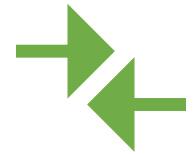
```
Program porReferencia;

procedure uno (var num: integer);
Begin
     if (num = 7)  then
        num:= num + 1;
     write (num);
end;
var
    x: integer;
begin
  x:= 7;
  uno (x);
end.
```

Dentro del procedimiento uno, el parámetro **num** comparte la dirección de memoria con **x** (variable del programa)

¿Cómo funciona?

AyPI – **MODULARIZACIÓN**

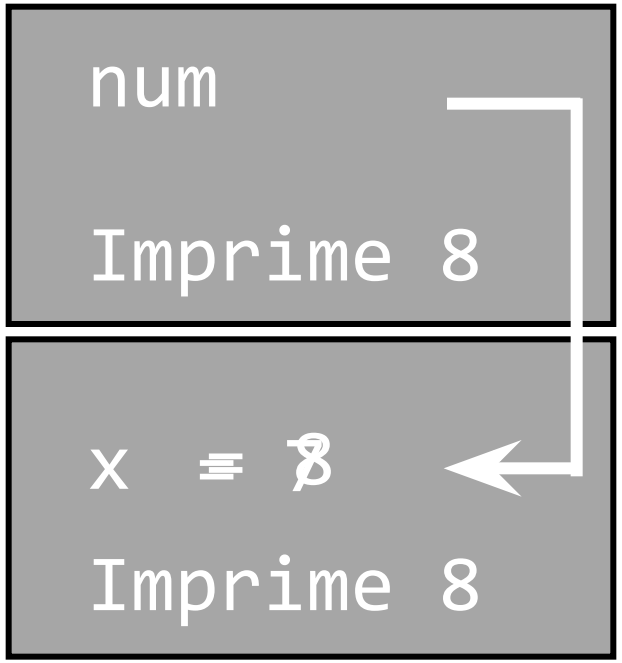
COMUNICACIÓN



```
Program porReferencia;

procedure uno (var num: integer);
Begin
    if (num = 7)  then
       num:= num + 1;
    write (num);
end;
var
   x: integer;
begin
  x:= 7;
  uno (x);
  write(x);
end.
```

Procedimiento uno
Variables locales
Parámetros

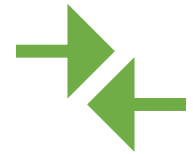
Programa ppal
Variables globales
Variables de prog



MEMORIA

AyPI – **MODULARIZACIÓN**

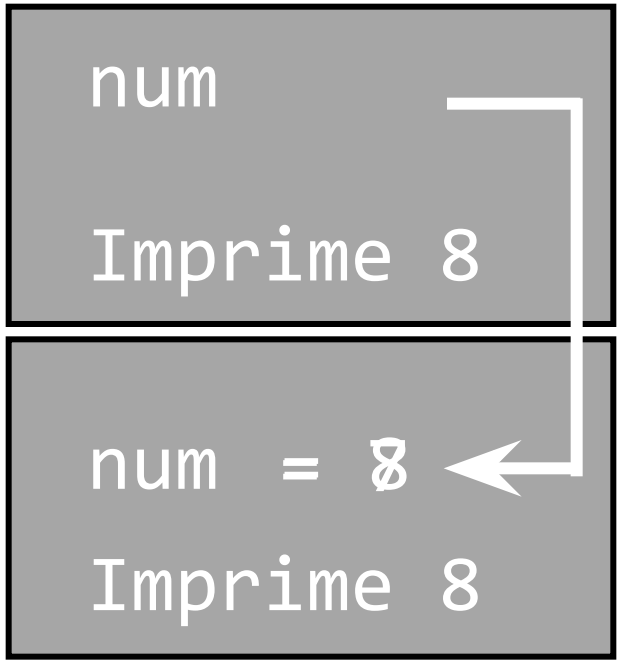
COMUNICACIÓN



```
Program porReferencia;

procedure uno (var num: integer);
Begin
    if (num = 7)  then
        num:= num + 1;
    write (num);
end;
var
   num: integer;
begin
  num:= 7;
  uno (num);
  write(num);
end.
```

Procedimiento uno
Variables locales
Parámetros

Programa ppal
Variables globales
Variables de prog



MEMORIA

AyPI – **MODULARIZACIÓN**

COMUNICACIÓN

- El número y tipo de los argumentos utilizados en la invocación a un módulo deben coincidir con el número y tipo de parámetros del encabezamiento del módulo.
- Un parámetro por valor debiera ser tratado como una variable de la cual el módulo hace una copia y la utiliza localmente. Algunos lenguajes permiten la modificación local de un parámetro por valor, pero toda modificación realizada queda en el módulo en el cual el parámetro es utilizado.
- El número y tipo de los argumentos utilizados en la invocación a un módulo deben coincidir con el número y tipo de parámetros del encabezamiento del módulo.

AyPI – **MODULARIZACIÓN**

COMUNICACIÓN

¿Es correcto?

```
Program uno;

Var
    x:integer;
    c:char;

procedure ejemplo (var a:integer; j:char);
begin
  // código del procedimiento ejemplo
end;

begin
     c:=‘a’;
     ejemplo (15,c);
end.
```

AyPI – **MODULARIZACIÓN**

COMUNICACIÓN

¿Es correcto?

```
Program uno;

Var
    x:integer;

procedure ejemplo (var a:integer; j:char);
begin
  // código del procedimiento ejemplo
end;

begin
   x:=25;
   ejemplo (x,'p');
end.
```

AyPI – **MODULARIZACIÓN**

COMUNICACIÓN

Escriba un programa que lea un número entero e imprima si el número es primo o no. Realice dos soluciones:

a) Modularice utilizando un procedimiento
b) Modularice utilizando una función

17  Imprime True

- ¿Qué es lo que se modulariza?
- ¿Qué parámetros necesita el módulo?
- ¿Cómo descompongo el número?

AyPI – MODULARIZACIÓN

COMUNICACIÓN

```
program uno;
procedure esNumPrimo (num:integer; var resultado:boolean);
var
  …
begin
   …
end;

Var
 valor:integer; primo: boolean;
Begin
   read (valor);
   esNumPrimo(valor, primo);
   write (primo);
End.
```

Copia el valor de valor

Comparte la dirección con sum

Cómo se implementa esNumPrimo?

AyPI – **MODULARIZACIÓN**

COMUNICACIÓN

```
procedure esNumPrimo (num:integer; var resultado:boolean);
var
 divisor:integer;

Begin
  resultado:= true;
  divisor:= num div 2;
  while (divisor > 1) and (resultado) do
   if num mod divisor = 0 then
      resultado:= false
   else
      divisor:= divisor - 1;
end;
```

Copia el valor recibido

Devuelve si es primo o no

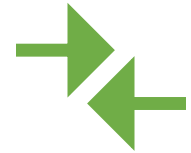
Asumo de entrada que el número es primo

¿Y con una función?

Encontré que num es divisible por El valor actual de divisor. Por lo tanto num no es primo.

AyPI – **MODULARIZACIÓN**

COMUNICACIÓN



```
function esNumPrimo (num:integer): boolean;
var
  divisor:integer;
  resultado: boolean;
Begin
  resultado:= true;
  divisor:= num div 2;
  while (divisor > 1) and (resultado) do
   if num mod divisor = 0 then
     resultado:= false
   else
      divisor:= divisor - 1;
  esNumPrimo:= resultado;
end;
```

Tipo de datos que devuelve

Copia el valor recibido

¿Por qué uso una variable resultado?

¿Cómo escribo el programa?

AyPI – MODULARIZACIÓN

COMUNICACIÓN

```
program uno;
function esNumPrimo (num:integer):boolean;
var
 ...
begin
  ...
end;

Var
 valor:integer; primo: boolean;
Begin
  read (valor);
  primo:= esNumPrimo (valor);
  write (primo);
End.
```

¿Otra forma de invocar a la función esNumPrimo?

AyPI – MODULARIZACIÓN

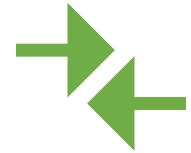
COMUNICACIÓN

```
program uno;
function esNumPrimo (num:integer):boolean;
var
  ...
begin
   ...
end;

Var
 valor:integer;
Begin
   read (valor);
   write (esNumPrimo (valor));
End.
```

¿Qué modifico si quiero leer números hasta leer el número 0, y para cada uno evaluar si es primo?

AyPI – **MODULARIZACIÓN**

COMUNICACIÓN



```
program uno;
function esNumPrimo (num:integer):boolean;
begin
  ...
end;

Var
 valor:integer;
Begin
  read (valor);
  while (valor <> 0) do
    begin
      write (esNumPrimo (valor));
      read(valor);
    end;
End.
```

AyPI – **MODULARIZACIÓN**

COMUNICACIÓN

Escriba un programa que lea números enteros hasta leer el valor 50. Para cada número par leído informar el dígito de la unidad. Al finalizar informa la unidad más baja y la alta entre todos los números leídos.

4367 → No imprime nada por ser impar

24 → Imprime 4

1382 → Imprime 2

50 → Termina Imprime 'La unidad más alta es 4, la mas baja es 2'

- ¿Qué es lo que se modulariza?
- ¿Con qué tipo de módulos?

AyPI – **MODULARIZACIÓN**

COMUNICACIÓN

Escriba un programa que lea números enteros hasta leer el valor 50. Para cada número par leído informar el dígito de la unidad.

4367 → No imprime nada por ser impar

24 → Imprime 4

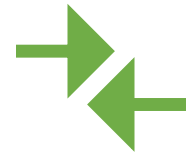
1382 → Imprime 2

50 → Termina, imprime
‘La unidad más alta es 4,
la más baja es 2’

```
Leo un número (num)
While (num <> 50) do
 begin
   if (el nro leído es par) then
     determino la unidad
      actualizo la unidad más alta y
            mas baja
   leo un número (num)
end.
```

AyPI – **MODULARIZACIÓN**

COMUNICACIÓN



Opción 1

```
function esPar (n:integer): boolean;
Var
  ok:boolean;
Begin
   if (n MOD 2 = 0) then ok:= true
   else ok:= false;
   esPar:= ok;
end;
```

Opción 2

```
function esPar (n:integer): boolean;
Var
 ok:Boolean;
Begin
   ok:= (n MOD 2 = 0);
   esPar:= ok;
end;
```

Opción 3

```
function esPar(n:integer):boolean;
Begin
    esPar:= (n MOD 2 = 0);
end;
```

AyPI – **MODULARIZACIÓN**

COMUNICACIÓN

```
program uno;
function esPar (n:integer):boolean;
 begin
   esPar:= (n MOD 2 = 0);
 end;

var
 num:integer;  par:boolean;
begin
  read (num);
  while (num <> 50) do
   begin
     par:= esPar(num);
     if (par = true) then
       determino la unidad
       actualizo la unidad más alta y más baja
     read(num);
   end;
 end.
```

AyPI – **MODULARIZACIÓN**

COMUNICACIÓN

```
program uno;
function esPar (n:integer):boolean;
 begin
   esPar:= (n MOD 2 = 0);
 end;

var
 num:integer;
begin
   read (num);
   while (num <> 50) do
    begin
      if (esPar(num) = true) then
        determino la unidad
        actualizo la unidad más alta y más baja
      read(num);
    end;
 end.
```

Para determinar la unidad ¿qué elijo: una función o un procedimiento?

AyPI – **MODULARIZACIÓN**

COMUNICACIÓN

```
Procedure unidad( num:integer; var uni:integer);
Begin
    uni:= num MOD 10;
End;
```

```
function unidad( num:integer):integer;
Begin
    unidad:= num MOD 10;
End;
```

AyPI – **MODULARIZACIÓN**

COMUNICACIÓN

```
Procedure maxmin( unidad:integer; var max,min:integer);

Begin
   if (unidad > max) then
        max:= unidad;
    if (unidad < min) then
        min:= unidad;
End;
```

```
function maximo(unidad, max:integer):integer;
Begin
  if (unidad > max) then
       max:= unidad;
  maximo:= max;
End;


function minimo(unidad, min:integer):integer;
Begin
  if (unidad < min) then
       min:= unidad;
  minimo:= min;
End;
```

AyPI – **MODULARIZACIÓN**

```
program MaxMinVer1;
Var num,maxu,minu,uni:integer;
Begin
   read (num);
   maxu := -1; minu := 9999;
   while (num <> 50) do begin
      if (esPar (num)) then  begin
        uni:= unidad (num);
        maxmin(uni, maxu, minu);
        write(uni);
        end;
      read (num);
     end;
   Write (‘Unidad máx’, maxu);
   Write (‘Unidad mín’, minu);
end.
```

COMUNICACIÓN

```
program MaxMinVer2;
Var num,maxi,mini,uni:integer;
Begin
   read (num);
   maxi := -1; mini := 9999;
   while (num <> 50) do begin
        if (esPar (num) ) then begin
          uni:= unidad (num);
          maxi:= maximo(uni, maxi);
          mini:= minimo(uni, mini);
          write(uni);
        end;
        read (num);
     end;
   Write (‘Unidad máx’, maxi);
   Write (‘Unidad mín’, mini);
end.
```

AyPI – **MODULARIZACIÓN**

COMUNICACIÓN

```
Program MaxMinVer1;

function esPar (n:integer):boolean;
    begin
      esPar:= (n MOD 2 = 0);
    end;

function unidad( num:integer):integer;
    Begin
       unidad:= num MOD 10;
    End;


Procedure maxmin(unidad:integer; var max,min:integer);
Begin
   if (unidad > max) then max:= unidad;
   if (unidad < min) then min:= unidad;
End;
```

```
Var  num,maxu,minu:integer;
Begin
 read (num);
 maxu := -1; minu := 9999;
 while (num <> 50) do
     begin
      if (esPar (num)) then  begin
        uni:= unidad (num);
        maxmin(uni, maxu, minu);
        write(uni);
        end;
      read (num);
     end;
   Write (‘Dig máx’, maxu);
   Write (‘Dig mín’, minu);
end.
```

AyPI – **MODULARIZACIÓN**

EJERCICIO - 1

```
Program Ej1;
function esNumPrimo (num:integer):boolean;
    begin
    ...
    end;
Var
 encontrados, numero, cuantos:integer;

Begin
   encontrados:= 0;  numero:= 1;
   read(cuantos);
   while (encontrados < cuantos) do
      begin
      if (esNumPrimo(numero)) then
          begin
          writeln(numero, ‘ es primo.’);
          encontrados:= encontrados + 1;
          end;
      numero:= numero + 1;
      end;
End.
```

¿Qué hace este programa?

AyPI – **MODULARIZACIÓN**

EJERCICIO - 2

```
Program Ej2;
function esNumPrimo (num:integer):boolean;
   begin
   ...
   end;
Var
 numero, inferior, superior:integer;

Begin
   read(inferior);
   read(superior);
   for numero:= inferior to superior do
      begin
      if (esNumPrimo(numero)) then
          writeln(numero, ‘ es primo.’);
      end;
End.
```

¿Qué hace este programa?

AyPI – **MODULARIZACIÓN**

EJERCICIO - 3

```
Program Ej3;

function suma (a, b, c, d: integer): integer;
Begin
  suma:= a + b + c + d;
End;

function promedio(a, b, c, d: integer): real;
begin
  promedio:= suma(a, b, c, d) / 4;
end;

Var
 a,b,c,d:integer;
Begin
  read(a);  read(b);  read(c);  read(d);
  write ( promedio(a, b, c, d) );

  read(a);  read(b);  read(c);  read(d);
  write ( suma(a, b, c, d) );
End.
```

¿Qué hace este programa?