

TEMAS DE INTRO I

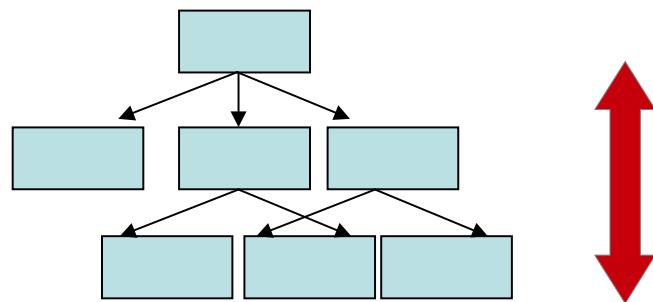
- Pilas (datos/estructuras de control)
- Filas (datos/estructuras de control)
- Modularización y Parámetros
- Variables
- **Funciones** y Método de Desarrollo
- Arreglos
- Matrices

MODULARIZACION

- Cada problema es dividido en un número de subproblemas más pequeños, cada uno de los cuales a su vez, puede dividirse en un conjunto de subproblemas más pequeños aún, y así siguiendo.
- Tareas generales se pueden pensar e implementar como módulos independientes a ser usados en distintos programas

Ventajas:

- Cada uno de estos subproblemas resulta más simple de resolver
- Reuso de módulos en distintos programas
- Abstracción



Implementación de módulos en Pascal

Pascal posee el concepto de rutina: serie de instrucciones con un nombre que puede ser invocada muchas veces a través del nombre. Pascal tiene dos formas de implementar rutinas:

- **PROCEDIMIENTO:** conjunto de instrucciones que ejecutan una tarea
- **FUNCIÓN:** conjunto de instrucciones que luego de evaluar varios parámetros, devuelve un resultado
 - Ambos pueden tener múltiples parámetros (en el caso de la función sólo son parámetros de entrada)

Program Ejemplo;

```
Function Triple (dato:integer):integer;  
begin  
    Triple:= dato *3  
end;  
var  
    dato, valor:integer;  
  
Begin {Del programa principal}  
.....  
end.
```

El nombre representa un cálculo

Siempre devuelve un valor
De un tipo primitivo

El nombre de la función debe estar asignado
a la izquierda. Debe haber una asignación en
cada camino alternativo del código.

Program Ejemplo;

```
Function Triple (dato:integer): integer;  
begin  
  Triple:= dato *3  
end;  
var  
  dato, valor:integer;
```

Begin {Del programa principal- EJEMPLOS de uso}

```
dato:=5;  
valor := Triple (dato);  
.....  
if Triple(dato) > 23 then  
  writeln ('El triple de' , dato, ' supera el 23');  
.....
```

```
Write(Triple(dato));  
.....
```

La función devuelve un resultado,
debe ser usado en cualquier instrucción
como si fuera una variable del tipo que devuelve

Triple(dato);

end.

¡ MAL INVOCADA!

PROCEDIMIENTO

Archivo Editar Buscar Ver Formato Lenguaje Configuración



PasaOrdenadoconProc.pas

```
1 Program
2
3   uses Estructu;
4   procedure NOMBREPROCEDIMIENTO();
5   begin
6
7   end;
8
9   var
10
11  begin
12    .....
13    .....
14    .....
15    .....
16    NOMBREPROCEDIMIENTO();
17    .....
18  end.
```

FUNCIONES

Archivo Editar Buscar Ver Formato Lenguaje Configuración



PasaOrdenadoconProc.pas

```
1 Program
2
3   uses Estructu;
4
5   Function TRIPLE(): integer
6   begin
7     .....
8     .....
9     .....
10    TRIPLE:= .....
11  end;
12
13  var
14
15  begin
16    .....
17    .....
18    .....
19    .....
20    IF TRIPLE() > 23 THEN.....
21    .....
22  end.
```

Procedimientos Funciones

Definición:

Procedure

Nombre de Acción

Sin tipo de Retorno

Sin asignación a su nombre

Definición:

Function

Nombre de Evaluación

CON tipo de Retorno

CON asignación a su nombre
para todos los caminos
posibles

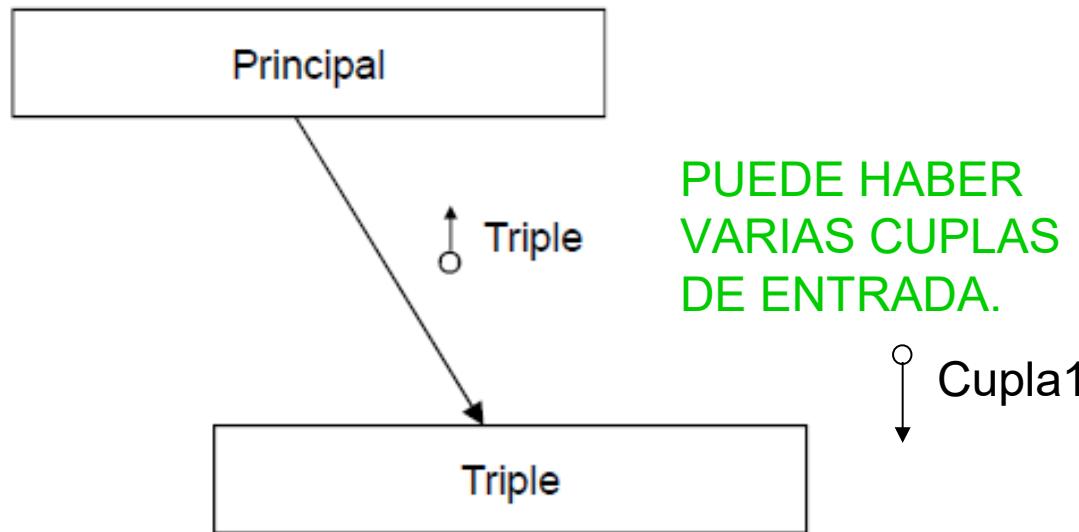
Invocación:

En sentencia completa para que
se ejecute.

Invocación:

Igual que una variable de su tipo,
excepto a la izquierda de una
asignación.

Funciones en el Diagrama de Estructura



Function **Triple** (.....NO PUEDE HABER PASAJE POR REFERENCIA.....): integer;

Implementar una función que reciba una pila y devuelva la cantidad de elementos que posee

```
Function CantElem (unaPila: Pila): integer;  
var  
    cant: integer;  
    descarte: Pila;  
  
begin  
    cant:=0;  
    inicPila(descarte, ' ');  
    while not pilaVacia(unaPila) do  
        begin  
            cant:= cant +1;  
            apilar (descarte, desapilar(unaPila))  
        end;  
    CantElem:= cant  
end;
```

Usando la función recién definida, realice un programa que dada dos pilas informe cuál tiene más cantidad de elementos.

ComparaPilas

CantElem

UnaPila

CantElem

```
program ComparaPilas;
{$INCLUDE /usr/Estructu}

Function CantElem (unaPila: Pila): integer;
.....
end;

var
  pila1, pila2:Pila;

begin
  ReadPila(pila1);
  ReadPila(pila2);
  If CantElem(Pila1) < CantElem(Pila2) then
    writeln('La Pila1 tiene menos elementos que la pila 2')
  else
    If CantElem(Pila1) > CantElem(Pila2) then
      writeln('La Pila1 tiene más elementos que la pila 2')
    else
      writeln('Ambas pilas tienen la misma cantidad de elementos')
end.
```

- Realice una función que recibe a un entero y a una pila como parámetros y devuelva Verdadero o Falso si la pila contiene a ese entero.

```
program estaEnPila;  
uses Estructu;
```

```
Function EstaElemento (unaPila:Pila; elemento:integer): boolean;  
Var  
    descarte: Pila;  
Begin  
    inicPila(descarte);  
    while not pilaVacia(unaPila) and (tope(unaPila) <> elemento) do  
        apilar (descarte, desapilar(unaPila));  
    if pilaVacia(unaPila) then  
        EstaElemento:= false  
    else  
        EstaElemento:= true  
end;
```

```
program estaEnPila;  
{$INCLUDE /usr/Estructu}
```

```
Function EstaElemento (unaPila:Pila; elemento:integer): boolean;
```

```
.....  
end;
```

```
var
```

```
    unapila:Pila;  
    elemento: integer;
```

```
begin
```

```
    ReadPila(unapila);
```

```
    ReadIn(elemento);
```

```
    if EstaElemento(unapila, elemento) then
```

```
        writeln('El elemento ', elemento, ' está')
```

```
    else
```

```
        writeln('El elemento ', elemento, ' no está')
```

```
end.
```

Forma de Trabajo



- 1- Comprender el problema**
- 2- Estudiar todas las posibles situaciones**
- 3- Plantear todas las estrategias posibles**
- 4- Seleccionar la más adecuada**
- 5- Formalizar la solución mediante el diagrama de Estructura**
- 6- Codificar**
- 7- Validar la Solución según las situaciones planteadas en el punto 2**

Forma de Trabajo

- 1- Comprender el problema**
- 2- Estudiar todas las posibles situaciones**
- 3- Plantear todas las estrategias posibles**
- 4- Seleccionar la más adecuada**
- 5- Formalizar la solución mediante el diagrama de Estructura**
- 6- Codificar**
- 7- Validar la Solución según las situaciones planteadas en el punto 2**

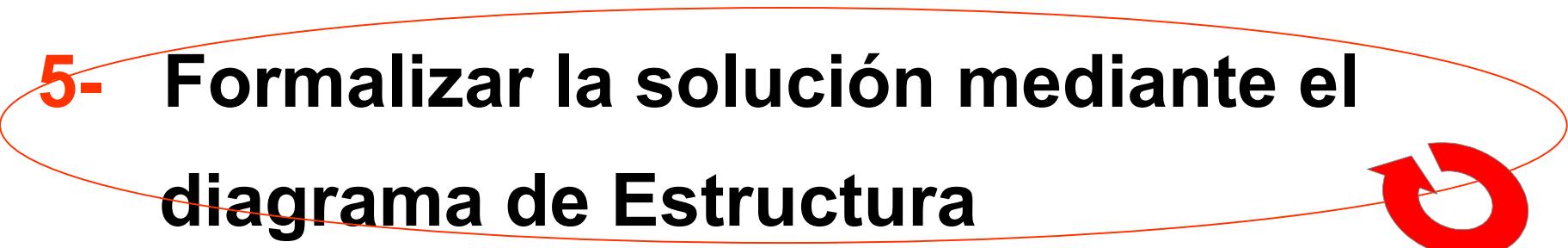
Forma de Trabajo

- 1- Comprender el problema**
- 2- Estudiar todas las posibles situaciones**
- 3- Plantear todas las estrategias posibles**
- 4- Seleccionar la más adecuada**
- 5- Formalizar la solución mediante el diagrama de Estructura**
- 6- Codificar**
- 7- Validar la Solución según las situaciones planteadas en el punto 2**

Forma de Trabajo

- 1- Comprender el problema**
- 2- Estudiar todas las posibles situaciones**
- 3- Plantear todas las estrategias posibles**
- 4- Seleccionar la más adecuada**
- 5- Formalizar la solución mediante el diagrama de Estructura**
- 6- Codificar**
- 7- Validar la Solución según las situaciones planteadas en el punto 2**

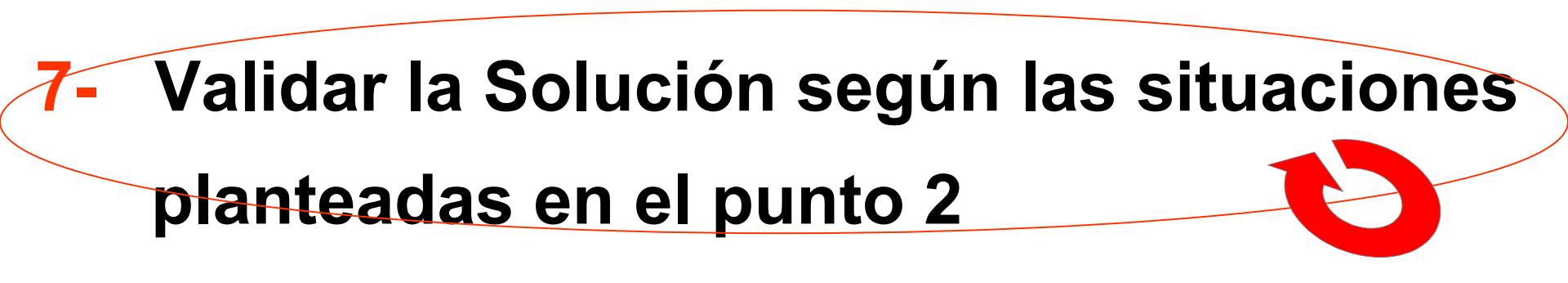
Forma de Trabajo

- 1- Comprender el problema**
 - 2- Estudiar todas las posibles situaciones**
 - 3- Plantear todas las estrategias posibles**
 - 4- Seleccionar la más adecuada**
 - 5- Formalizar la solución mediante el diagrama de Estructura**
 - 6- Codificar**
 - 7- Validar la Solución según las situaciones planteadas en el punto 2**
- 

Forma de Trabajo

- 1- Comprender el problema**
 - 2- Estudiar todas las posibles situaciones**
 - 3- Plantear todas las estrategias posibles**
 - 4- Seleccionar la más adecuada**
 - 5- Formalizar la solución mediante el diagrama de Estructura**
 - 6- Codificar**
 - 7- Validar la Solución según las situaciones planteadas en el punto 2**
- 

Forma de Trabajo

- 1- Comprender el problema**
 - 2- Estudiar todas las posibles situaciones**
 - 3- Plantear todas las estrategias posibles**
 - 4- Seleccionar la más adecuada**
 - 5- Formalizar la solución mediante el diagrama de Estructura**
 - 6- Codificar**
 - 7- Validar la Solución según las situaciones planteadas en el punto 2**
- 

Realice un programa que determina si todos los elementos de una pila PARTE están incluidos en una Pila MODELO, sin importar el orden.

Forma de Trabajo

Enunciado:

Realice un programa que determina si todos los elementos de una pila PARTE están incluidos en una Pila MODELO, sin importar el orden.

1- Comprender el problema

Forma de Trabajo

Enunciado:

Realice un programa que determina si todos los elementos de una pila PARTE están incluidos en una Pila MODELO, sin importar el orden.

2- Estudiar todas las posibles situaciones

Que debe pasar si:

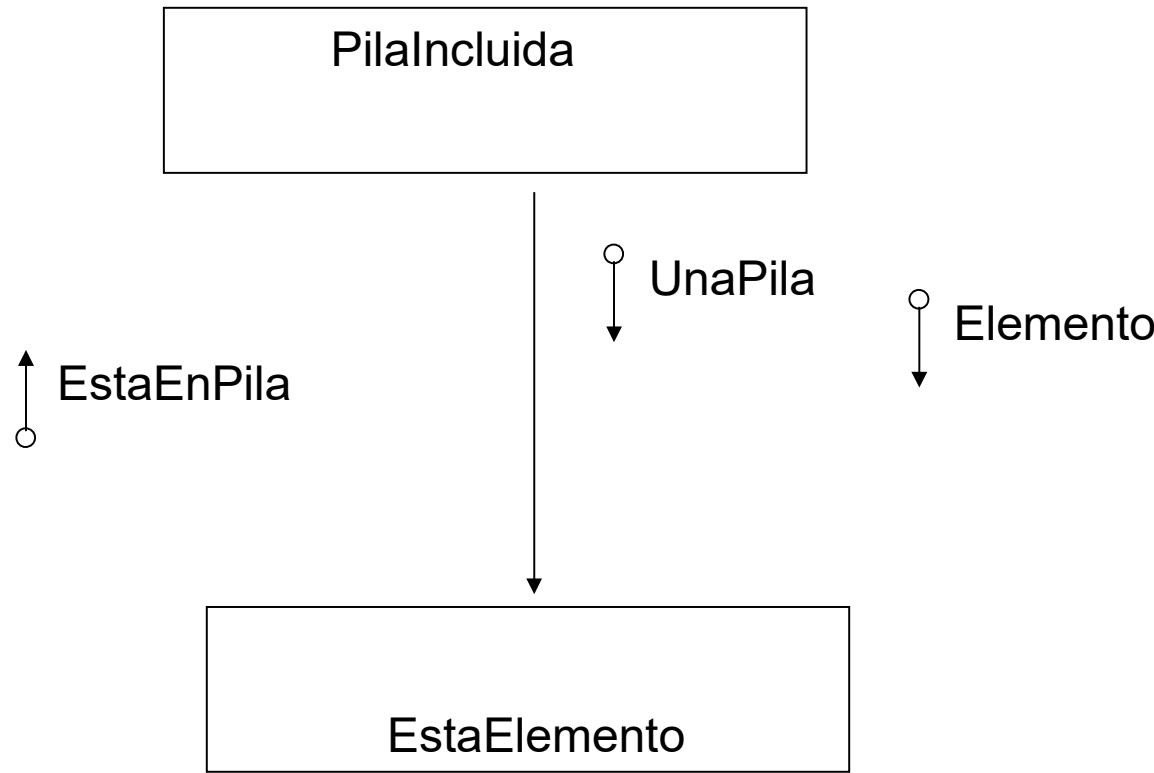
- una de las pilas vacías?
 - ambas vacías?
 - la primera con más elementos?
 - la segunda con más elementos?
 - ambas igual cantidad de elementos?
-

Forma de Trabajo

Realice un programa que determina si todos los elementos de una pila PARTE están incluidos en una Pila MODELO, sin importar el orden.

- 3- Planpear todas las estrategias posibles**
- 4- Seleccionar la más adecuada**

Forma de Trabajo



5- Formalizar la Estrategia con un diagrama de Estructura

Forma de Trabajo

6. Codificar

```
program estaEnPila;  
{$INCLUDE /usr/Estructu}
```

```
Function EstaElemento (unaPila:Pila; elemento:integer): boolean;
```

```
Var
```

```
descarte: Pila;
```

```
Begin
```

```
inicPila(descarte, ' ' );
```

```
while not pilaVacia(unaPila) and (tope(unaPila) <> elemento) do  
    apilar (descarte, desapilar(unaPila));
```

```
if pilaVacia(unaPila) then      EstaElemento:= false  
                                else      EstaElemento:= true;
```

```
end;
```

```
program estaEnPila;  
{$INCLUDE /usr/Estructu}  
Function EstaElemento (unaPila:Pila; elemento:integer): boolean;  
.....  
End;  
  
var  
    unapila, incluida, descarte:Pila;  
Begin  
    ReadPila(unapila);  
    writeln('Ingrese Pila a verificar');  
    ReadPila(incluida);  
    While not pilaVacia(incluida) and estaElemento(unapila, tope(incluida)) do  
        apilar (descarte, desapilar(incluida));  
    if pilaVacia(incluida)  
        then writeln(' esta incluida')  
        else writeln(' NO esta incluida')  
end.
```

Forma de Trabajo

7- Validar la Solución según las situaciones planteadas en el punto 2

Realizar ejecuciones con las siguientes situaciones evaluando que el programa funciones según lo esperado

- una de las pilas vacías
- ambas vacías
- la primera con más elementos
- la segunda con más elementos
- ambas con igual cantidad de elementos