

TEMAS DE INTRO I

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

DATOS y VARIABLES

Program PasaPila;

{Este Programa pasa los elementos de la pila Origen a Destino}

{\$INCLUDE /IntroProg/Estructu}

var

Origen, Destino: pila;

Las pilas **ORIGEN** y **DESTINO** son los datos de este programa

Begin

ReadPila(Origen);

InicPila(Destino, '');

While not PilaVacia(Origen) do

 Apilar (Destino, Desapilar(Origen));

WritePila(Origen);

WritePila(Destino)

end.

¿Cuáles son los datos en este programa?

- Los **datos** son toda aquella información con los que opera el programa (¡Recordar ingredientes de una receta!)
- Un **DATO** se asocia a una **VARIABLE** del programa (ej. Origen, Destino, Auxiliar)

```
Program PasaPila;  
{.....}  
  
{$INCLUDE /IntroProg/Estructu}
```

```
Var    Origen, Destino, Auxiliar: Pila;  
  
.....
```

Una **VARIABLE** tiene dos componentes

NOMBRE: es la forma de identificar a la variable, cada vez que se la referencia se lo hace por su nombre. Ej: Origen, Destino, Auxiliar, Descarte

TIPO: define los posibles valores que puede tomar y el conjunto de operaciones que se puede hacer sobre ella. (ej. PILA y FILA)

Program PasaPila;

{\$INCLUDE /IntroProg/Estructu}

var

Origen: Pila;

NOMBRE DE LA VARIABLE

TIPO DE LA VARIABLE

Qué instrucciones se pueden realizar con la variable Origen?

ReadPila(Origen);

InicPila(Origen, '3 6 7 ')

PilaVacia(Origen)

Tope(Origen)

Apilar(Origen, Desapilar(nombreOtra Pila))

WritePila(Origen)

ReadFila(Origen);

InicFila(Origen, '3 6 7 ')

FilaVacia(Origen)

Primero(Origen)

Agregar(Origen, Eliminar(nombreOtra Pila))

WriteFila(Origen)

TIPOS

TIPO: define el conjunto de posibles valores que puede tomar una variable y las operaciones que se puede hacer sobre ella. Ej: PILA y FILA

Una clasificación

ESTRUCTURADOS

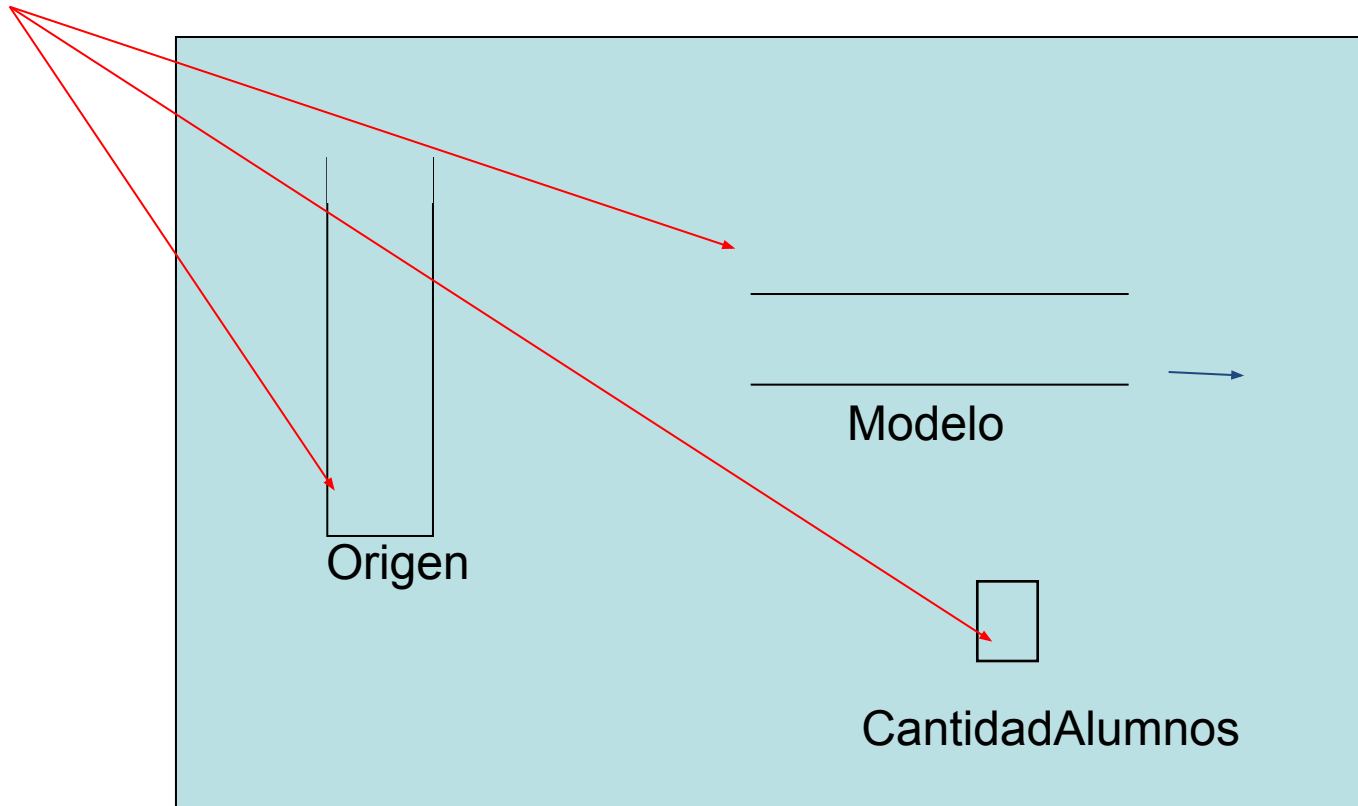
- Permiten almacenar un conjunto de elementos con una determinada organización. (ej. TIPO PILAS define una organización “apilada” de elementos y el conjunto de operaciones que se puede realizar)

NO ESTRUCTURADOS

- Permiten almacenar un UNICO elemento. (por ejemplo una variable de tipo INTEGER)

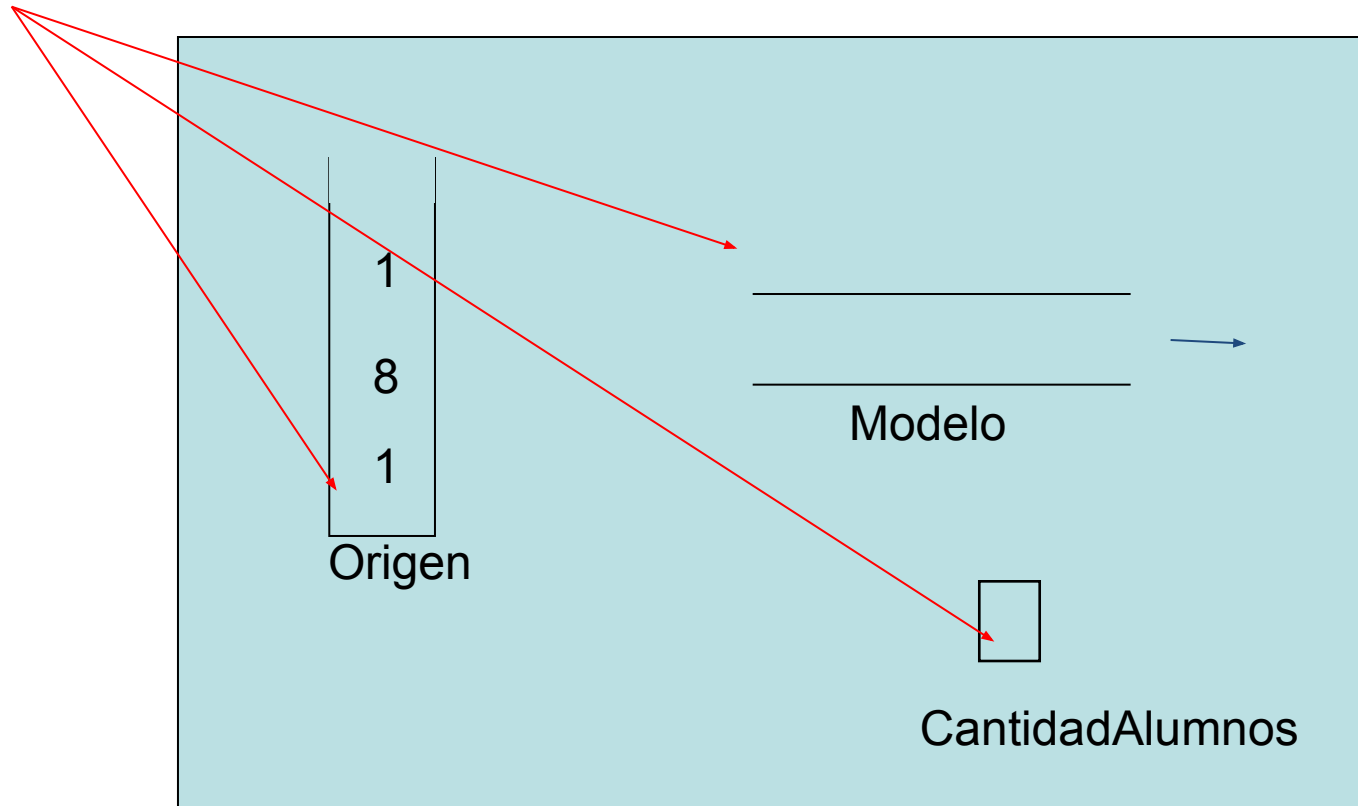
Variables en memoria

!!!!SE DEBEN INICIALIZAR!!!!



Variables en memoria

!!!!SE DEBEN INICIALIZAR!!!!



Var

Origen: Pila;

Begin

InicPila(Origen, ' ');

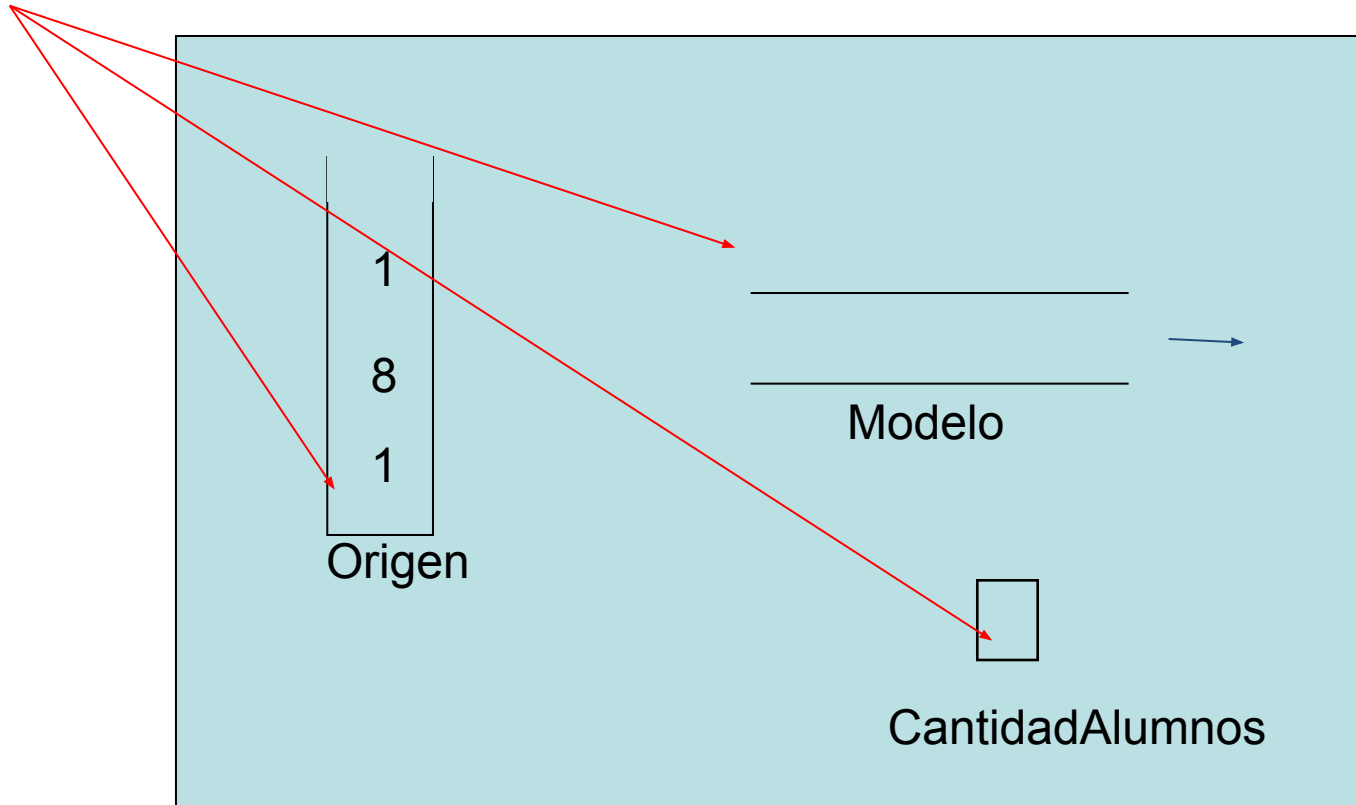
Apilar (Origen, 1);

Apilar (Origen,8);

Apilar (Origen,1);

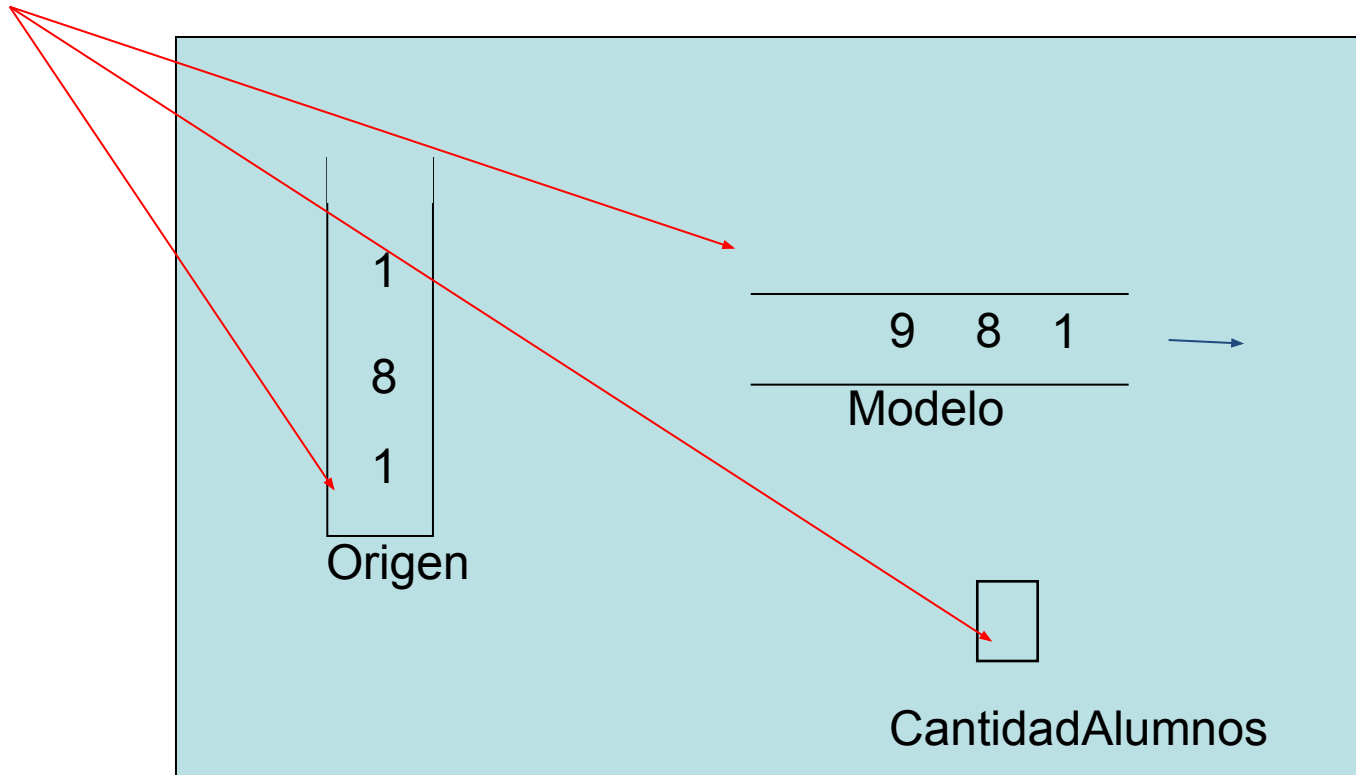
Variables en memoria

!!!!SE DEBEN INICIALIZAR!!!!



Variables en memoria

!!!!SE DEBEN INICIALIZAR!!!!



Var

Modelo: Fila;

Begin

InicFila(Origen, ' ');

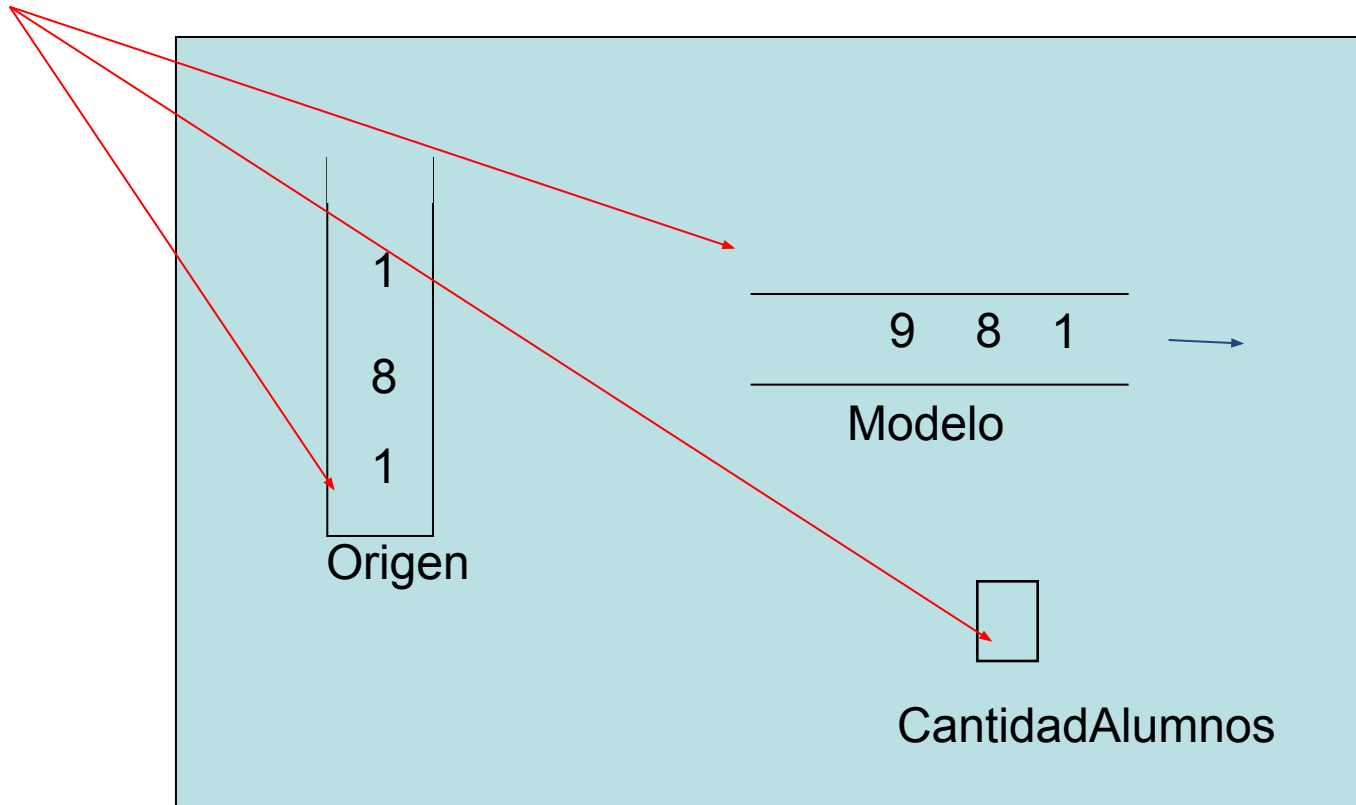
Agregar (Modelo, 1);

Agregar (Modelo, 8);

Agregar (Modelo, 9);

Variables en memoria

!!!!SE DEBEN INICIALIZAR!!!!

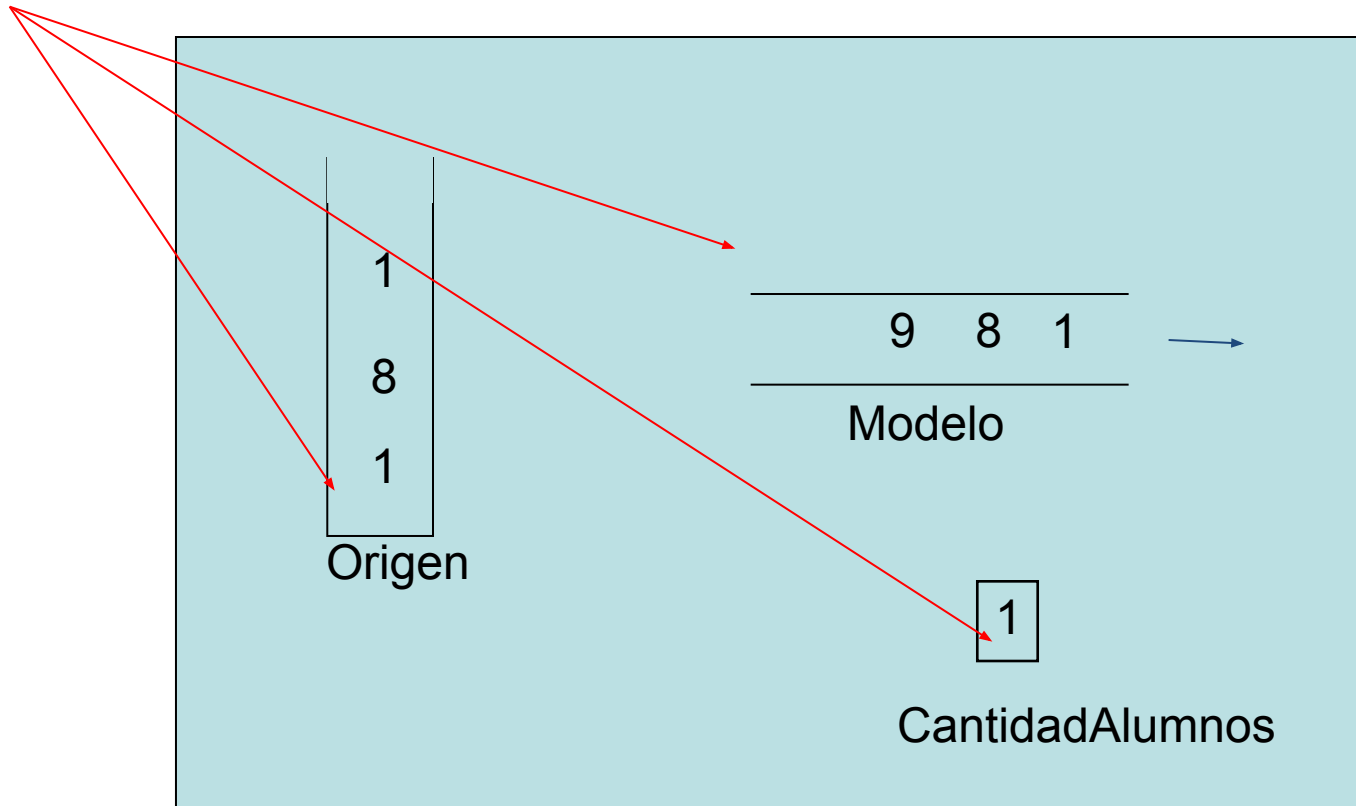


Var

CantidadAlumnos: integer;

Variables en memoria

!!!!SE DEBEN INICIALIZAR!!!!

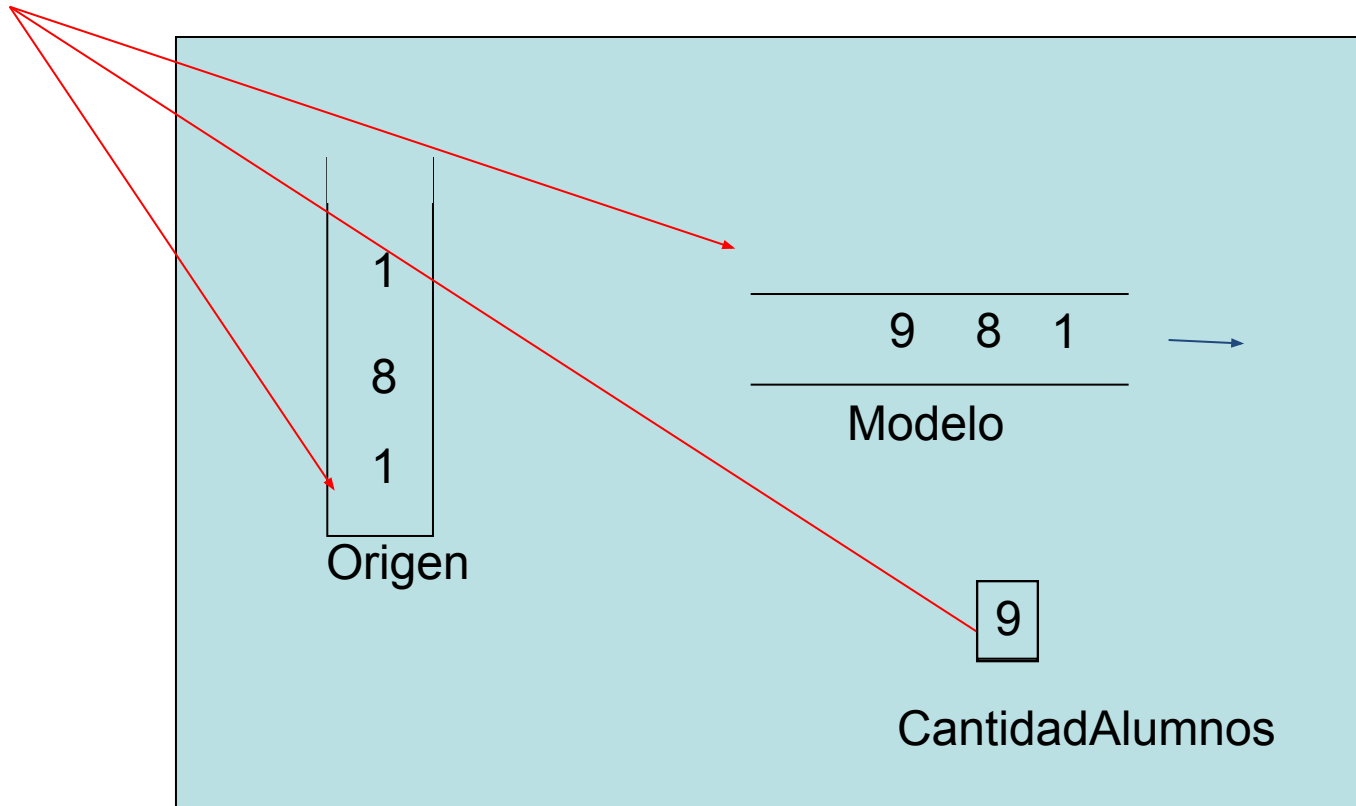


```
Var  
CantidadAlumnos: integer;  
begin  
  CantidadAlumnos:=1;
```

Quiero que CantidadAlumnos sea 1

Variables en memoria

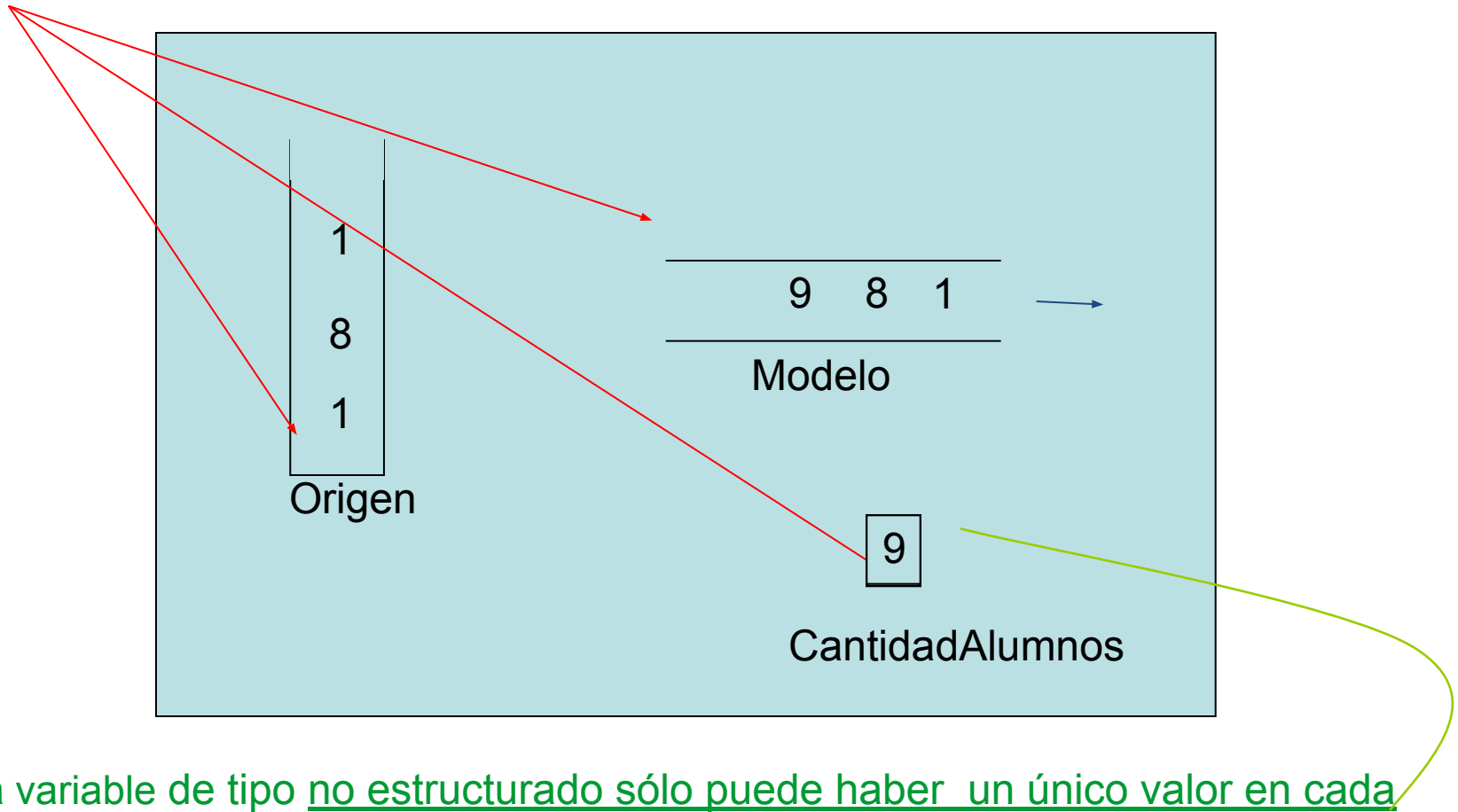
!!!!SE DEBEN INICIALIZAR!!!!



Quiero que CantidadAlumnos sea 9 \longrightarrow `CantidadAlumnos := 9;`

Variables en memoria

!!!!SE DEBEN INICIALIZAR!!!!



En una variable de tipo no estructurado sólo puede haber un único valor en cada instante.

Si SE CAMBIA DE VALOR SE PISA EL EXISTENTE

Asignación de valores a una variable de tipo predefinido

PROGRAM EJEMPLO;

VAR EDAD: integer;

BEGIN

EDAD := 22;

MEMORIA

BASURA

EDAD

Asignación de valores a una variable de tipo predefinido

PROGRAM EJEMPLO;

VAR EDAD: integer;

BEGIN

EDAD := 22;

OPERADOR DE ASIGNACIÓN

MEMORIA

22

EDAD

TIPOS no estructurados predefinidos

NOMBRE

VALORES

OPERACIONES

INTEGER

REAL

BOOLEAN

CHAR

TIPOS no estructurados predefinidos

NOMBRE	VALORES	OPERACIONES
INTEGER	- 32768 a 32767	* DIV MOD + - ()

$$(2 * 3) + 4$$

$$6 + 4$$

$$2 * (3 + 4)$$

TIPOS no estructurados predefinidos

NOMBRE	VALORES	OPERACIONES
INTEGER	- 32768 a 32767	* DIV MOD + - ()

6 DIV 2 = 3
6 MOD 2 = 0

7 DIV 2 = 3
7 MOD 2 = 1

X MOD 2 = 0 ☐ X ES PAR

TIPOS no estructurados predefinidos

NOMBRE	VALORES	OPERACIONES
INTEGER	- 32768 a 32767	* DIV MOD + - ()
REAL	1 E-38 a 1 E+38	* / + - ()

134,456778

TIPOS no estructurados predefinidos

NOMBRE	VALORES	OPERACIONES
INTEGER	- 32768 a 32767	* DIV MOD + - ()
REAL	1 E-38 a 1 E+38	* / + - ()
BOOLEAN	TRUE FALSE	NOT AND OR ()

NOT PILAVACIA(origen)

TIPOS no estructurados predefinidos

NOMBRE	VALORES	OPERACIONES
INTEGER	- 32768 a 32767	* DIV MOD + - ()
REAL	1 E-38 a 1 E+38	* / + - ()
BOOLEAN	TRUE FALSE	NOT AND OR ()
CHAR	'A' 'Z', 'a' .. 'z', '0' , '1' '9', '?', '*', ' ', etc.	+ (concatena)

Asignación de valores a una variable de tipo predefinido

PROGRAM EJEMPLO;

VAR EDAD: integer;

BEGIN

EDAD := 22;

EDAD := EDAD + 1;

MEMORIA

BASURA

EDAD

Asignación de valores a una variable de tipo predefinido

PROGRAM EJEMPLO;

VAR EDAD: integer;

BEGIN

EDAD := 22;

EDAD := EDAD + 1;

MEMORIA

22

EDAD

Asignación de valores a una variable de tipo predefinido

```
PROGRAM EJEMPLO;
```

```
VAR  EDAD: integer;
```

```
BEGIN
```

```
    EDAD := 22;
```

```
    EDAD := EDAD + 1;
```

MEMORIA

23

EDAD

Asignación de valores a una variable de tipo predefinido

PROGRAM EJEMPLO;

VAR EDAD: integer;

BEGIN

EDAD := 22;

EDAD := EDAD + 1;

OPERADOR DE ASIGNACIÓN

MEMORIA

23

EDAD

Asignación de valores a una variable de tipo predefinido

PROGRAM EJEMPLO;

VAR EDAD: integer;

BEGIN

EDAD := 22;

EDAD := EDAD + 1;

SENTENCIA DE ASIGNACIÓN

MEMORIA

23

EDAD

SENTENCIA DE ASIGNACIÓN

VARIABLE **:=** EXPRESIÓN ;

¡Deben ser de igual tipo!

```
VAR  Edad: integer;  
     Origen: Pila;  
     Edad := 1;  
     Edad := 'P';  
     Origen := 1 3 4 );
```

Program Ejemplo,

var

edadMayor, edadMenor : integer;

promedio: real;

diaLindo, llueve, sol: boolean;

BEGIN

edadMenor:= 11;

edadMayor:= edadMenor + 1;

promedio:= (edadMenor + edadMayor) / 2;

llueve:= false;

sol:= true;

diaLindo := not llueve and sol;

.....

END.

El lugar de memoria que se asigna a cada variable depende del tipo.

Program Ejemplo,
var

```
edadMayor, edadMenor : integer;  
promedio: real;  
diaLindo, llueve, sol: boolean;
```

BEGIN

```
edadMenor:= 11;  
edadMayor:= edadMenor + 1;  
promedio:= (edadMenor + edadMayor) / 2;  
llueve:= false;  
sol:= true;  
diaLindo := not llueve and sol;
```

.....
END.

BASURA

EdadMayor

BASURA

EdadMenor

Program Ejemplo,

var

edadMayor, edadMenor : integer;

promedio: real;

diaLindo, llueve, sol: boolean;

BEGIN

edadMenor:= 11;

edadMayor:= edadMenor + 1;

promedio:= (edadMenor + edadMayor) / 2;

llueve:= false;

sol:= true;

diaLindo := not llueve and sol;

.....

END.

BASURA

EdadMayor

BASURA

promedio

BASURA

EdadMenor

Program Ejemplo,
var

edadMayor, edadMenor : integer;
promedio: real;

diaLindo, llueve, sol: boolean;

BEGIN

edadMenor:= 11;
edadMayor:= edadMenor + 1;
promedio:= (edadMenor + edadMayor) / 2;
llueve:= false;
sol:= true;
diaLindo := not llueve and sol;

.....
END.

BASURA

EdadMayor

BASURA

EdadMenor

BASURA

promedio



diaLindo



llueve



sol

Program Ejemplo,
var

edadMayor, edadMenor : integer;
promedio: real;
diaLindo, llueve, sol: boolean;

BEGIN

```
edadMenor:= 11;  
edadMayor:= edadMenor + 1;  
promedio:= (edadMenor + edadMayor) / 2;  
llueve:= false;  
sol:= true;  
diaLindo := not llueve and sol;
```

.....
END.

BASURA

EdadMayor

11

EdadMenor

BASURA

promedio



diaLindo



llueve



sol

Program Ejemplo,
var

edadMayor, edadMenor : integer;
promedio: real;
diaLindo, llueve, sol: boolean;

BEGIN

edadMenor:= 11;
edadMayor:= edadMenor + 1;
promedio:= (edadMenor + edadMayor) / 2;
llueve:= false;
sol:= true;
diaLindo := not llueve and sol;

.....
END.

12

EdadMayor

11

EdadMenor

BASURA

promedio



diaLindo



llueve



sol

Program Ejemplo,
var

edadMayor, edadMenor : integer;
promedio: real;
diaLindo, llueve, sol: boolean;

BEGIN

edadMenor:= 11;
edadMayor:= edadMenor + 1;
promedio:= (edadMenor + edadMayor) / 2;
llueve:= false;
sol:= true;
diaLindo := not llueve and sol;

.....
END.

12

EdadMayor

11

EdadMenor

11,5

promedio



diaLindo



llueve



sol

Program Ejemplo,
var

edadMayor, edadMenor : integer;
promedio: real;
diaLindo, llueve, sol: boolean;

BEGIN

edadMenor:= 11;
edadMayor:= edadMenor + 1;
promedio:= (edadMenor + edadMayor) / 2;
llueve:= false;
sol:= true;
diaLindo := not llueve and sol;

.....
END.

12

EdadMayor

11

EdadMenor

11,5

promedio



diaLindo

F

llueve



sol

Program Ejemplo,
var

edadMayor, edadMenor : integer;
promedio: real;
diaLindo, llueve, sol: boolean;

BEGIN

edadMenor:= 11;
edadMayor:= edadMenor + 1;
promedio:= (edadMenor + edadMayor) / 2;
llueve:= false;
sol:= true;
diaLindo := not llueve and sol;

.....
END.

12

EdadMayor

11

EdadMenor

11,5

promedio



diaLindo

F

llueve

T

sol

Program Ejemplo,
var

edadMayor, edadMenor : integer;
promedio: real;
diaLindo, llueve, sol: boolean;

BEGIN

edadMenor:= 11;
edadMayor:= edadMenor + 1;
promedio:= (edadMenor + edadMayor) / 2;
llueve:= false;
sol:= true;
diaLindo := not llueve and sol;

.....
END.

12

EdadMayor

11

EdadMenor

11,5

promedio

T

diaLindo

F

llueve

T

sol

Las variables de tipo char, integer y real puede ser leídas o mostradas por pantalla

Las variables de tipo boolean No pueden ser leídas o mostradas por pantalla

Program Ejemplo,
var

edad: integer;
promedio: real;
letra: char;
diaLindo: boolean;

BEGIN

Readln(edad);
Readln(promedio);
Readln(letra);
Readln(dialindo);

.....

writeln(edad);
writeln(promedio);
writeln(letra);
writeln(dialindo);

Problema

Hacer un programa que calcula la suma de números que va ingresando el usuario por pantalla. El usuario informa cuando no quiere cargar más. Al finalizar informa el resultado.

Nota: no modularizar

Program SumarNumeros;

var
 SigueCargando: Char;
 Numero, SumaNumeros: Integer;

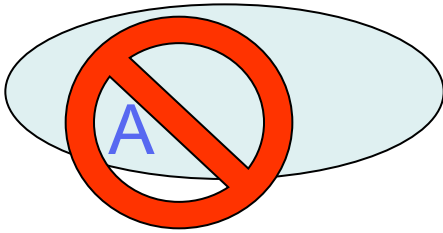
begin
SumaNumeros:= 0;
write ('¿Desea cargar numeros(ingrese s para si)?');
readln(SigueCargando);
While (SigueCargando = 's') do begin
 write (' Ingrese un número:');
 readln(Numero);
 SumaNumeros := SumaNumeros + Numero;
 write ('¿Desea seguir cargando(ingrese s para si)?');
 readln(SigueCargando);
end;
write (SumaNumeros)
end.

ALCANCE DE LAS VARIABLES

```
Program EJEMPLO;  
Procedure ProcUNO ( BX );  
Var  
    C: integer;  
Begin  
    .....A.....  
    .....BX.....  
    .... C .....  
end;  
  
....  
Var A B  
Begin  
    ProcUNO (B)
```

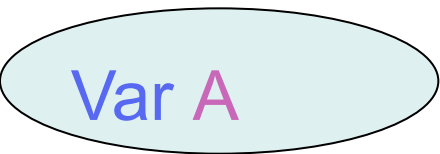
ALCANCE DE LAS VARIABLES

```
Program EJEMPLO;  
Procedure ProcUNO;  
Begin
```



```
end;
```

```
....
```



```
Begin  
ProcUNO
```

¡NO USAR!

VARIABLES
GLOBALES

ALCANCE DE LAS VARIABLES

Program EJEMPLO;

Procedure ProcUNO (B);

.....

.....

Begin

.....

..... B

.....

end;

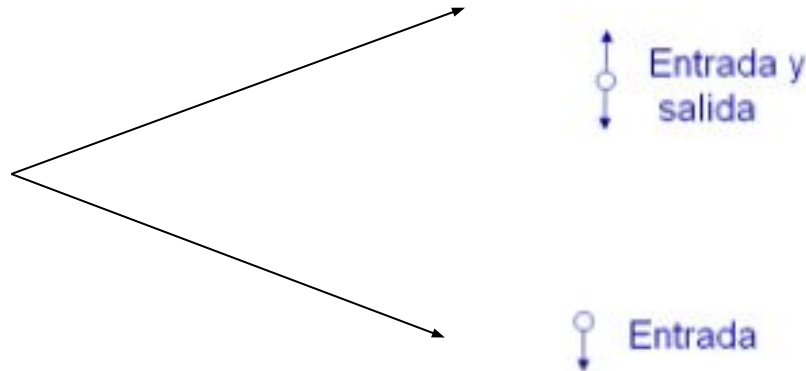
....

Var B

Begin

ProcUNO (B)

Pasaje de Parámetros



ALCANCE DE LAS VARIABLES

Program EJEMPLO;

Procedure ProcUNO (B);

.....

.....

Begin

.....

..... B

.....

end;

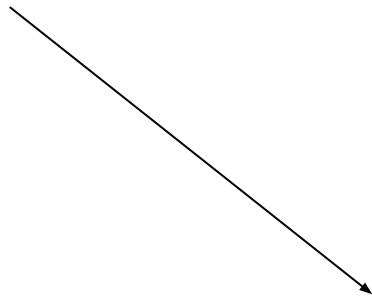
....

Var B

Begin

ProcUNO (B)

Pasaje de Parámetros



ALCANCE DE LAS VARIABLES

Program EJEMPLO;

Procedure ProcUNO (var B);

.....

.....

Begin

.....

..... B

.....

end;

....

Var B

Begin

ProcUNO (B)

Pasaje de Parámetros



Program EJEMPLO;

Procedure ProcUNO(dato:char)

Var

C: integer;

Begin

.....

C:= 10;

.....

end;

....

Var

dato: char

Begin

dato:= 's';

ProcUNO(dato) ;

Variables LOCALES a
los procedimientos

Program EJEMPLO;

Procedure ProcUNO (B);

Var C

.....

Begin

.....B.....

.....C.....

La definición de una variable como local o parámetro debe surgir del diagrama de Estructura

Si son cuplas del DE => parámetros

Si no están en el DE => son locales

end;

Problema

Pensar el ejercicio anterior(sumatoria de números) como un módulo que es invocado desde el programa principal y luego muestra por pantalla los resultados al usuario.

Principal



RealizarSuma

Principal



SumaNumeros

RealizarSuma

Program EjemploDeVariables;

...{ procedure }.....

var

SumaNumeros: Integer;

begin

SumaNumeros := 0;

RealizarSuma(SumaNumeros);

writeln(' Suma de números: ', SumaNumeros);

end.

```
Procedure RealizarSuma( var SumaNumeros: Integer);
```

```
var
```

```
    SigueCargando: Char;
```

```
    Numero: Integer;
```

```
begin
```

```
SigueCargando:= 's';
```

```
While (SigueCargando = 's') do begin
```

```
    write (' Ingrese un número:');
```

```
    readln(Numero);
```

```
    SumaNumeros := SumaNumeros + numero;
```

```
    write ('¿Desea seguir cargando?');
```


```
    readln( SigueCargando);
```

```
end;
```

```
end;
```

¿Puedo definir como variable local a una variable cuyo valor debo retornar? 

Se envía como parámetro entrada/ salida (por referencia)

¿Puedo definir como parámetro a una variable que sólo la necesito para trabajar localmente en el procedimiento? 

Se define como variable local al procedimiento