

TIPO DE DATOS:

Define el conjunto de posibles valores que puede tomar una variable y las operaciones que se puede hacer sobre ella. Ej: PILA, FILA, tipos simples primitivos

Una clasificación de tipos

NO ESTRUCTURADOS:

Permiten guardar un solo valor (tipos primitivos: integer, real, boolean, char)

ESTRUCTURADOS:

Conjunto de elementos organizados de acuerdo a una estructura (Pilas, Filas).

Un tipo de datos define las características de los elementos y las operaciones que se pueden hacer (por ejemplo Pila en Pascal almacena enteros y se puede desapilar, apilar...)

Arreglos

Arreglo de Enteros (capacidad 8)

8	9	3	10	6	7	9	7
1	2	3	4	5	6	7	8

Posición ó Índice

Arreglo de Char (capacidad 4)

a	b	v	t
1	2	3	4

Posición ó Índice

Es un Tipo de Datos

ESTRUCTURADO

HOMOGÉNEO

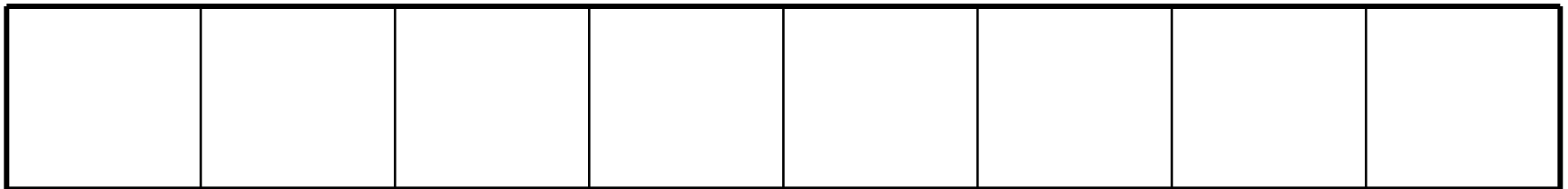
ESTÁTICO

INDEXADO

Arreglos en Pascal

Var **Nombre**: Array [**inicio**..**fin**] of **tipo Primitivo**;

Cantidad Estática y definida



Inicio

Sig Inicio

.....

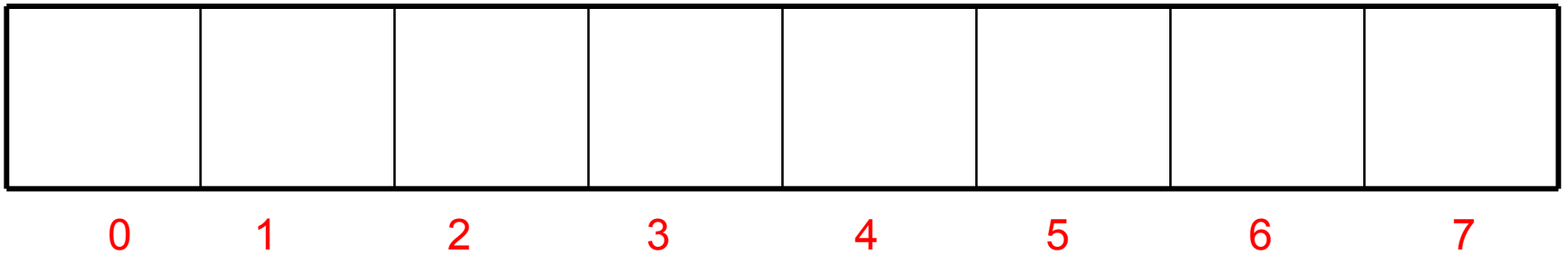
Ant fin

Fin

Arreglos en Pascal

Var **Nombre**: Array [**0..7**] of **tipo Primitivo**;

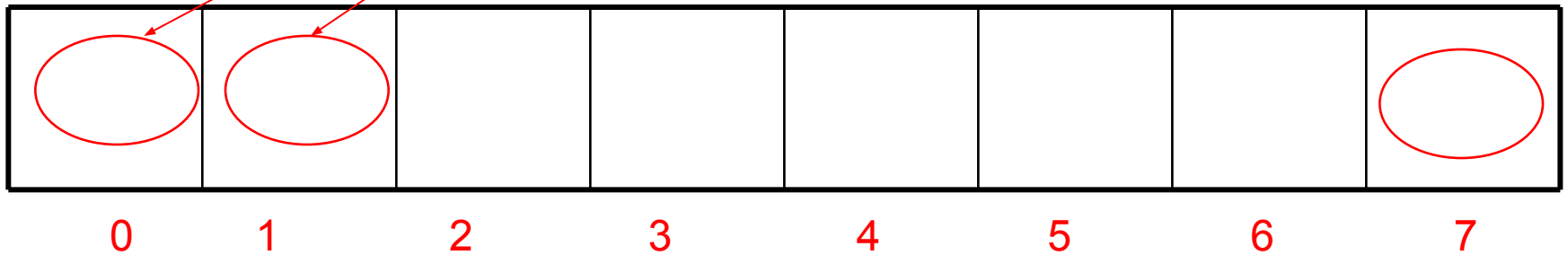
Cantidad Estática y definida



Arreglos en Pascal

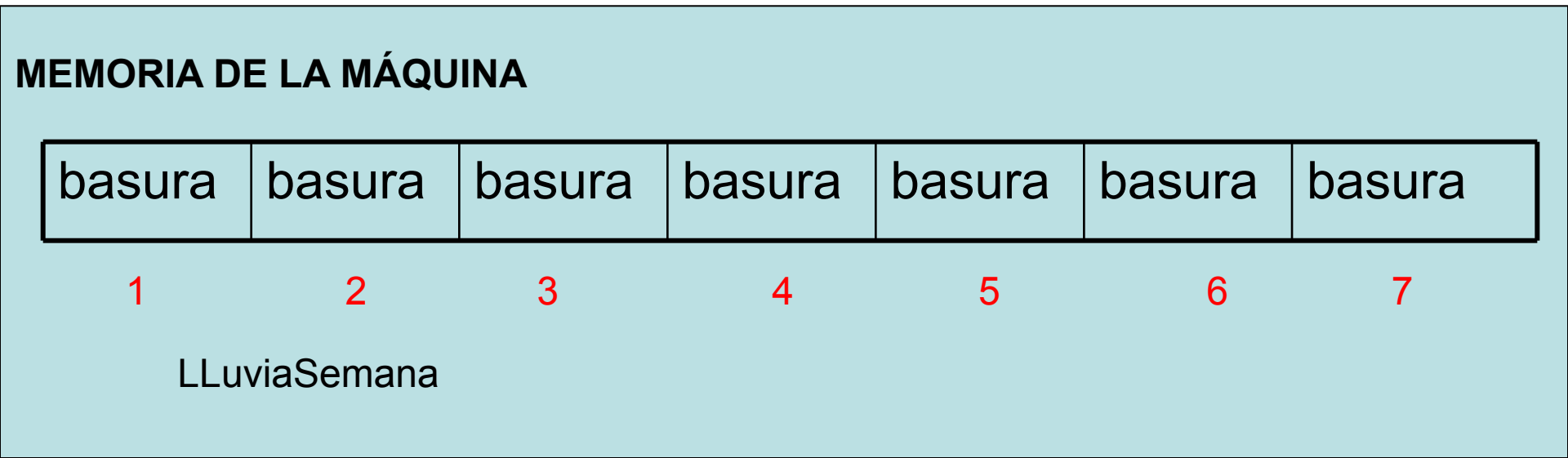
Var **Nombre**: Array [**0..7**] of **tipo Primitivo**;

Integer
Real
Boolean
char



Arreglos en Pascal

Var LluviaSemana: **Array** [1..7] of Integer;



Arreglos en Pascal

Var LluviaSemana: **Array** [1..7] of Integer

Begin

 LLuviaSemana[3]:=27;

MEMORIA DE LA MÁQUINA

basura	basura	27	basura	basura	basura	basura
--------	--------	-----------	--------	--------	--------	--------

1

2

3

4

5

6

7

LLuviaSemana

Arreglos en Pascal

Var LluviaSemana: **Array** [1..7] of Integer

Begin

LLuviaSemana[3]:=27;

LLuviaSemana[2]:=10;

MEMORIA DE LA MÁQUINA

basura	10	27	basura	basura	basura	basura
--------	----	----	--------	--------	--------	--------

1

2

3

4

5

6

7

LLuviaSemana

Arreglos en Pascal

Var LluviaSemana: **Array** [1..7] of Integer

Begin

LLuviaSemana[3]:=27;

LLuviaSemana[2]:=10;

LLuviaSemana[1]:=15 + LLuviaSemana[2];

MEMORIA DE LA MÁQUINA

25	10	27	basura	basura	basura	basura
----	----	----	--------	--------	--------	--------

1

2

3

4

5

6

7

LLuviaSemana

Arreglos en Pascal

Var LluviaSemana: **Array** [1..7] of Integer

Begin

LLuviaSemana[3]:=27;

LLuviaSemana[2]:=10;

LLuviaSemana[1]:=15 + LLuviaSemana[2];

LLuviaSemana[4]:= LLuviaSemana[2] + LLuviaSemana[3];

MEMORIA DE LA MÁQUINA

25	10	27	37	basura	basura	basura
----	----	----	----	--------	--------	--------

1

2

3

4

5

6

7

LLuviaSemana

Arreglos en Pascal

```
Var LluviaSemana: Array [1..7] of Integer
```

```
Begin
```

```
  LLuviaSemana[3]:=27;
```

```
  LLuviaSemana[2]:=10;
```

```
  LLuviaSemana[1]:=15 + LLuviaSemana[2];
```

```
  LLuviaSemana[4]:= LLuviaSemana[2] + LLuviaSemana[3];
```

```
  WriteIn(LLuviaSemana[1]);
```

MEMORIA DE LA MÁQUINA

25	10	27	37	basura	basura	basura
----	----	----	----	--------	--------	--------

1

2

3

4

5

6

7

LLuviaSemana

Control de LLenado

Como los arreglos son estructuras de longitud estática (predefinida e invariable en toda la ejecución del programa) debo asegurarme cuando utilizo el valor de una celda que la misma haya sido inicializada por el programa ya que puede contener valor “Basura”

-3865	Error	-50	-1	Error	28	0
1	2	3	4	5	6	7

LluviaSemanal

Control de llenado

- Cargar un valor **discernible** en TODAS las celdas
- Si la utilización de las celdas es en forma consecutiva, se puede usar una **frontera**, simulando una estructura dinámica

Control de llenado: valor Discernible

Inicializar TODAS las celdas del Arreglo con un valor “discernible”

Pasos:

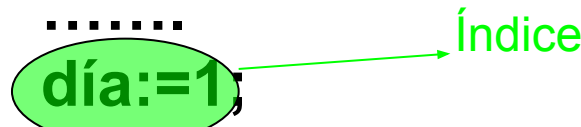
- 1) Seleccionar un valor discernible correctamente (0? -1? Cual?) Regla: valor inválido
- 2) Llenar todo el arreglo con ese valor
- 3) Condicionar la utilización de celdas a sólo aquellas que no tengan ese valor

Control de llenado: valor Discernible

Var LluviaSemanal: Array [1..7] of Integer

- Se elige el -1 como valor discernible.
- Al comenzar se inicializa todo el arreglo en -1

```
.....  
día:=1;  
While día < 8 do  
  Begin  
    LluviaSemanal[día] := -1;  
    día:= día +1  
  end;  
.....
```



- Estructura de control FOR

```
.....  
FOR dia:=1 TO 7 DO  LluviaSemanal[dia] := -1;
```

```
.....  
día:=1;  
While  día < 8  do  
    Begin  
        LluviaSemanal[día] := -1;  
        día:= día +1  
    end;
```

```
.....
```


Volviendo al control de llenado con un valor discernible

```
FOR dia :=1 to 7 DO LluviaSemanal[dia] := -1;
```

.....

LluviaSemanal

-1	-1	-1	-1	-1	-1	-1
----	----	----	----	----	----	----

1

2

3

4

5

6

7

Volviendo al control de llenado con un valor discernible

FOR dia :=1 **to** 7 **DO** LluviaSemanal[dia] := -1;

.....

LluviaSemanal[2] := 6;

LluviaSemanal[3] := 7;

LluviaSemanal[5] := 17;

LluviaSemanal[6] := 9;

LluviaSemanal

-1	6	7	-1	17	9	-1
----	---	---	----	----	---	----

1

2

3

4

5

6

7

Volviendo al control de llenado con un valor discernible

```
FOR dia :=1 to 7 DO LluviaSemanal[dia] := -1;
```

.....

```
REadln(LluviaSemanal[2] );
```

```
LluviaSemanal[3] := 7;
```

```
LluviaSemanal[5] := 17;
```

```
LluviaSemanal[6] := 9;
```

```
For dia := 1 to 7 do
```

Sólo se utilizan las celdas cuyo valor es $\neq -1$

```
  if LluviaSemanal[dia]  $\neq$  -1 then
```

```
    writeln(LluviaSemanal[dia]);
```

.....

LluviaSemanal

-1	6	7	-1	17	9	-1
----	---	---	----	----	---	----

1

2

3

4

5

6


7

□ Control de llenado posición límite o Frontera

4	5	17	0	9	-1234	6
---	---	----	---	---	-------	---

LLuviaSemanal

↑
FRONTERA

- 
- 1- La forma de carga debe ser consecutivas sin espacios “vacíos”
 - 2- Definir una variable que marcará la FRONTERA
dividiendo la FRONTERA entre las celdas utilizadas y las no utilizadas.
 - 3- La variable FRONTERA debe ser utilizada tanto para agregar valores como para consultar los ya agregados.
 - 4- Se simula una estructura dinámica

-4	-5	17	0	9	-1234	6
----	----	----	---	---	-------	---

MaxDiaCargado

0

LluviaSemanal

Program EjemploArregloFrontera;

{ Este porción de programa muestra un ejemplo de manejo de arreglos con Frontera}

Var

LluviaSemanal: array [1..7] of integer;
día, MaxDiaCargado: Integer;

begin

MaxDiaCargado:=0; {Arreglo vacío, la FRONTERA está en cero}

MaxDiaCargado:= MaxDiaCargado +1; {se agrega un valor, entonces se debe correr
la FRONTERA}

LluviaSemanal[MaxDiaCargado]:= 33;

MaxDiaCargado:= MaxDiaCargado +1; {se agrega un valor, entonces se debe correr
la FRONTERA}

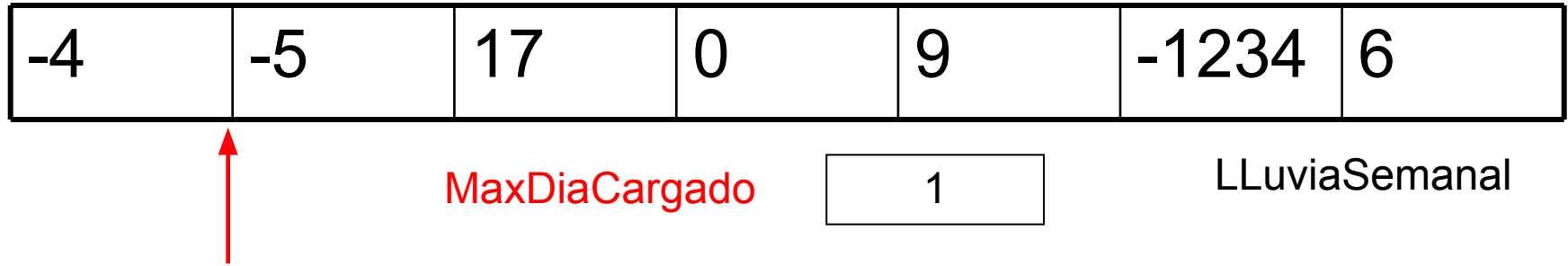
LluviaSemanal[MaxDiaCargado]:= 28;

{ se quiere imprimir, sólo se debe considerar los valores a la izquierda de la FRONTERA}

for día := 1 to MaxDiaCargado do

writeln (LluviaSemanal[día]);

end.



Program EjemploArregloFrontera;
{ Este porción de programa muestra un ejemplo de manejo de arreglos con Frontera}

Var
LLuviaSemanal: array [1..7] of integer;
día, MaxDiaCargado: Integer;

begin
MaxDiaCargado:=0; {Arreglo vacío, la FRONTERA está en cero}
MaxDiaCargado:= MaxDiaCargado +1; {se agrega un valor, entonces se debe correr
la FRONTERA}
LLuviaSemanal[MaxDiaCargado]:= 33;
MaxDiaCargado:= MaxDiaCargado +1; {se agrega un valor, entonces se debe correr
la FRONTERA}
LLuviaSemanal[MaxDiaCargado]:= 28;

{ se quiere imprimir, sólo se debe considerar los valores a la izquierda de la FRONTERA}
for día := 1 to MaxDiaCargado do
 writeln (LLuviaSemanal[día]);
end.

33	-5	17	0	9	-1234	6
----	----	----	---	---	-------	---



MaxDiaCargado

1

LLuviaSemanal

Program EjemploArregloFrontera;

{ Este porción de programa muestra un ejemplo de manejo de arreglos con Frontera}

Var

LluviaSemanal: array [1..7] of integer;

Dia, MaxDiaCargado: Integer;

begin

MaxDiaCargado:=0; {Arreglo vacío, la FRONTERA está en cero}

MaxDiaCargado:= MaxDiaCargado +1; {se agrega un valor, entonces se debe correr la FRONTERA}

LluviaSemanal[MaxDiaCargado]:= 33;

MaxDiaCargado:= MaxDiaCargado +1; {se agrega un valor, entonces se debe correr la FRONTERA}

LluviaSemanal[MaxDiaCargado]:= 28;

{ se quiere imprimir, sólo se debe considerar los valores a la izquierda de la FRONTERA}

for dia:= 1 **to** MaxDiaCargado **do**

writeln (LluviaSemanal[MaxDiaCargado]);

end.

33	-5	17	0	9	-1234	6
----	----	----	---	---	-------	---



MaxDiaCargado

2

LLuviaSemanal

Program EjemploArregloFrontera;

{ Este porción de programa muestra un ejemplo de manejo de arreglos con Frontera}

Var

LluviaSemanal: array [1..7] of integer;
día, MaxDiaCargado: Integer;

begin

MaxDiaCargado:=0; {Arreglo vacío, la FRONTERA está en cero}

MaxDiaCargado:= MaxDiaCargado +1; {se agrega un valor, entonces se debe correr
la FRONTERA}

LluviaSemanal[MaxDiaCargado]:= 33;

MaxDiaCargado:= MaxDiaCargado +1; {se agrega un valor, entonces se debe correr
la FRONTERA}

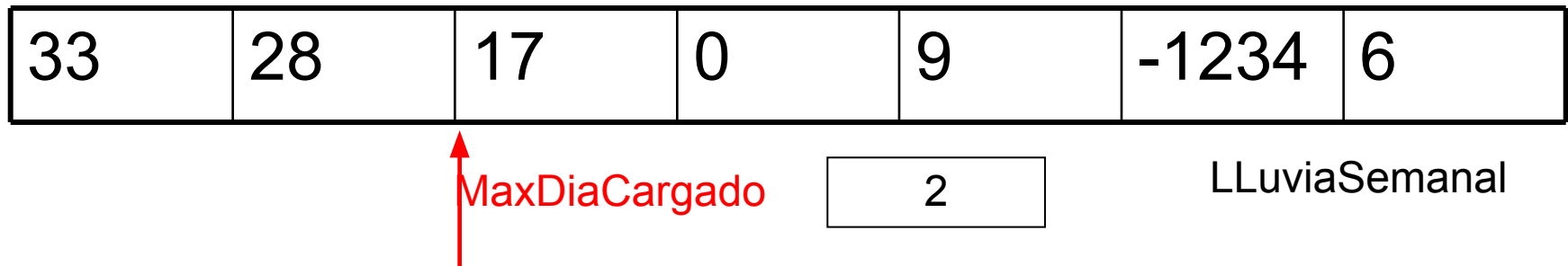
LluviaSemanal[MaxDiaCargado]:= 28;

{ se quiere imprimir, sólo se debe considerar los valores a la izquierda de la FRONTERA}

for día := 1 **to** MaxDiaCargado **do**

writeln (LluviaSemanal[día]);

end.



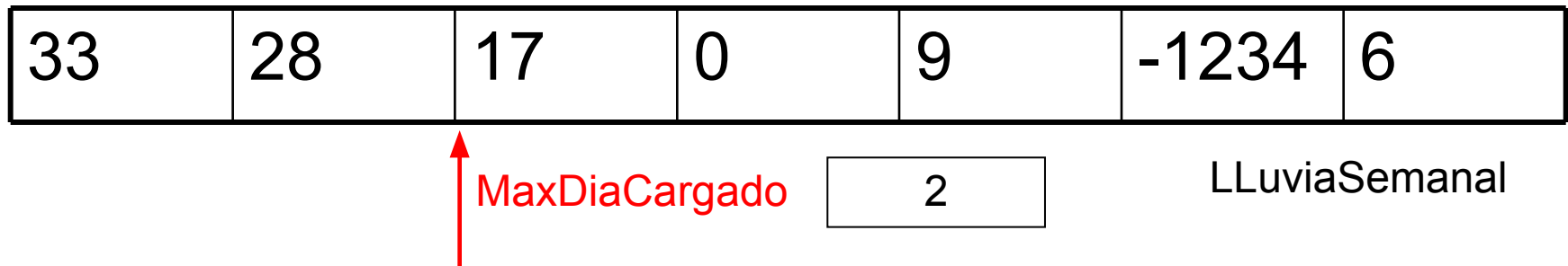
Program EjemploArregloFrontera;
{ Este porción de programa muestra un ejemplo de manejo de arreglos con Frontera}

Var
LLuviaSemanal: array [1..7] of integer;
día, MaxDiaCargado: Integer;

begin
MaxDiaCargado:=0; {Arreglo vacío, la FRONTERA está en cero}
MaxDiaCargado:= MaxDiaCargado +1; {se agrega un valor, entonces se debe correr la FRONTERA}
LLuviaSemanal[MaxDiaCargado]:= 33;
MaxDiaCargado:= MaxDiaCargado +1; {se agrega un valor, entonces se debe correr la FRONTERA}

LLuviaSemanal[MaxDiaCargado]:= 28;

{ se quiere imprimir, sólo se debe considerar los valores a la izquierda de la FRONTERA}
for día := 1 to MaxDiaCargado do
 writeln (LLuviaSemanal[día]);
end.



Program EjemploArregloFrontera;

{ Este porción de programa muestra un ejemplo de manejo de arreglos con Frontera}

Var

LLuviaSemanal: array [1..7] of integer;
 día, MaxDiaCargado: Integer;

begin

MaxDiaCargado:=0; {Arreglo vacío, la FRONTERA está en cero}

MaxDiaCargado:= MaxDiaCargado +1; {se agrega un valor, entonces se debe correr
 la FRONTERA}

LLuviaSemanal[MaxDiaCargado]:= 33;

MaxDiaCargado:= MaxDiaCargado +1; {se agrega un valor, entonces se debe correr
 la FRONTERA}

LLuviaSemanal[MaxDiaCargado]:= 28;

~~{ se quiere imprimir, sólo se debe considerar los valores a la izquierda de la FRONTERA}~~

~~for día := 1 to MaxDiaCargado do~~

~~writeln (LLuviaSemanal[día]);~~

~~end.~~

33	28	17	0	9	-1234	6
----	----	----	---	---	-------	---



MaxDiaCargado

2

LLuviaSemanal

Program EjemploArregloFrontera;

{ Este porción de programa muestra un ejemplo de manejo de arreglos con Frontera}

Var

LLuviaSemanal: array [1..7] of integer;
día, MaxDiaCargado: Integer;

begin

MaxDiaCargado:=0; {Arreglo vacío, la FRONTERA está en cero}

MaxDiaCargado:= MaxDiaCargado +1; {se agrega un valor, entonces se debe correr
la FRONTERA}

LLuviaSemanal[MaxDiaCargado]:= 33;

MaxDiaCargado:= MaxDiaCargado +1; {se agrega un valor, entonces se debe correr
la FRONTERA}

LLuviaSemanal[MaxDiaCargado]:= 28;

{ se quiere imprimir, sólo se debe considerar los valores a la izquierda de la FRONTERA}

for día := 1 **to** MaxDiaCargado **do**

writeln (LLuviaSemanal[día]);

end.

33

33	28	17	0	9	-1234	6
----	----	----	---	---	-------	---



MaxDiaCargado

2

LLuviaSemanal

Program EjemploArregloFrontera;

{ Este porción de programa muestra un ejemplo de manejo de arreglos con Frontera}

Var

LluviaSemanal: array [1..7] of integer;
día, MaxDiaCargado: Integer;

begin

MaxDiaCargado:=0; {Arreglo vacío, la FRONTERA está en cero}

MaxDiaCargado:= MaxDiaCargado +1; {se agrega un valor, entonces se debe correr
la FRONTERA}

LluviaSemanal[MaxDiaCargado]:= 33;

MaxDiaCargado:= MaxDiaCargado +1; {se agrega un valor, entonces se debe correr
la FRONTERA}

LluviaSemanal[MaxDiaCargado]:= 28;

~~{ se quiere imprimir, sólo se debe considerar los valores a la izquierda de la FRONTERA}~~

for día := 1 to MaxDiaCargado do

writeln (LluviaSemanal[día]);

end.

33

33	28	17	0	9	-1234	6
----	----	----	---	---	-------	---



MaxDiaCargado

2

LLuviaSemanal

Program EjemploArregloFrontera;

{ Este porción de programa muestra un ejemplo de manejo de arreglos con Frontera}

Var

LluviaSemanal: array [1..7] of integer;
día, MaxDiaCargado: Integer;

begin

MaxDiaCargado:=0; {Arreglo vacío, la FRONTERA está en cero}

MaxDiaCargado:= MaxDiaCargado +1; {se agrega un valor, entonces se debe correr
la FRONTERA}

LluviaSemanal[MaxDiaCargado]:= 33;

33

MaxDiaCargado:= MaxDiaCargado +1; {se agrega un valor, entonces se debe correr
la FRONTERA}

28

LluviaSemanal[MaxDiaCargado]:= 28;

{ se quiere imprimir, sólo se debe considerar los valores a la izquierda de la FRONTERA}

for día:= 1 **to** MaxDiaCargado **do**

writeln (LluviaSemanal[día]);

end.

33	28	17	0	9	-1234	6
----	----	----	---	---	-------	---



MaxDiaCargado

2

LLuviaSemanal

Program EjemploArregloFrontera;

{ Este porción de programa muestra un ejemplo de manejo de arreglos con Frontera}

Var

LluviaSemanal: array [1..7] of integer;

Dia, MaxDiaCargado: Integer;

begin

MaxDiaCargado:=0; {Arreglo vacío, la FRONTERA está en cero}

MaxDiaCargado:= MaxDiaCargado +1; {se agrega un valor, entonces se debe correr
la FRONTERA}

LluviaSemanal[MaxDiaCargado]:= 33;

33

MaxDiaCargado:= MaxDiaCargado +1; {se agrega un valor, entonces se debe correr
la FRONTERA}

28

LluviaSemanal[MaxDiaCargado]:= 28;

{ se quiere imprimir, sólo se debe considerar los valores a la izquierda de la FRONTERA}

for dia:= 1 to MaxDiaCargado do

writeln (LluviaSemanal[MaxDiaCargado]);

end.

33	28	17	0	9	-1234	6
----	----	----	---	---	-------	---



MaxDiaCargado

2

LLuviaSemanal

Program EjemploArregloFrontera;

{ Este porción de programa muestra un ejemplo de manejo de arreglos con Frontera}

Var

LLuviaSemanal: array [1..7] of integer;
día, MaxDiaCargado: Integer;

begin

MaxDiaCargado:=0; {Arreglo vacío, la FRONTERA está en cero}

MaxDiaCargado:= MaxDiaCargado +1; {se agrega un valor, entonces se debe correr
la FRONTERA}

LLuviaSemanal[MaxDiaCargado]:= 33;

33

MaxDiaCargado:= MaxDiaCargado +1; {se agrega un valor, entonces se debe correr
la FRONTERA}

28

LLuviaSemanal[MaxDiaCargado]:= 28;

{ se quiere imprimir, sólo se debe considerar los valores a la izquierda de la FRONTERA}

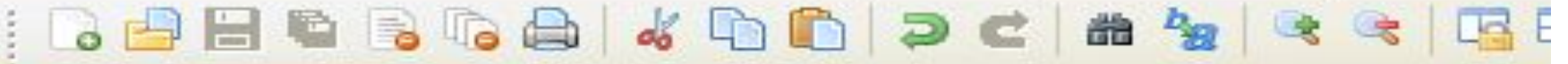
for día := 1 **to** MaxDiaCargado **do**

writeln (LLuviaSemanal[día]);

end.

C:\fpc\bin\EjemploSinConstantes.pas - Notepad++

Archivo Editor Buscar Ver Formato Lenguaje Configurar Macro Ejecutar Text



EjemploSinConstantes.pas

```
1  Program    ejemploSinConstantes;
2
3  var
4      LluviaSemanal: array [1..7] of Integer;
5      Dia: integer;
6
7  Begin
8      for dia:=1 to 7 do LluviaSemanal[dia]:=-1;
9
10     {distintas operaciones con el arreglo}
11
12     for dia:=1 to 7 do
13         if LluviaSemanal[dia] <> -1 then
14             Writeln( LluviaSemanal[dia]);
15     end.
16
17
```




EjemploConConstantes.pas

```
1  Program  ejemploConConstantes;
2  const
3      lunes=1;
4      domingo= 7;
5      sinValor = -1;
6  var
7      LluviaSemanal: array [lunes, domingo] of Integer;
8      Dia: integer;
9
10  Begin
11      for dia:=lunes to domingo do LluviaSemanal[dia]:= sinValor;
12
13      {distintas operaciones con el arreglo}
14
15      for dia:=lunes to domingo do
16          if LluviaSemanal[dia] <> sinValor then
17              Writeln( LluviaSemanal[dia]);
18      end.
19
20
```

Constantes

- Palabra reservada: **Const**
- Permite definir un literal (nombre, rótulo) y asignarle un valor que NO cambiara en todo el programa

Const

minimoAprobado=4;

- Permite LEGIBILIDAD y MODIFICABILIDAD

```
EjemploSinConstantes.pas
1  Program    ejemploSinConstantes;
2
3  var
4      LluviaSemanal: array [1..7] of Integer;
5      Dia: integer;
6
7  Begin
8      for dia:=1 to 7 do  LluviaSemanal[dia]:=-1;
9
10     {distintas operaciones con el arreglo}
11
12     for dia:=1 to 7 do
13         if LluviaSemanal[dia] < -1 then
14             Writeln( LluviaSemanal[dia]);
15     end.
```

```
EjemploConConstantes.pas
1  Program    ejemploConConstantes;
2  const
3      lunes=1;
4      domingo= 7;
5      sinValor = -1;
6  var
7      LluviaSemanal: array [lunes..domingo] of Integer;
8      Dia: integer;
9
10  Begin
11      for dia:= lunes to domingo do  LluviaSemanal[dia]:= sinValor;
12
13      {distintas operaciones con el arreglo}
14
15      for dia:= lunes to domingo do
16          if LluviaSemanal[dia] < sinValor then
17              Writeln( LluviaSemanal[dia]);
18  end.
```

program LluviaSemanal;

const

lunes=1;

domingo=7;

sinvalor=-1;

var

lluvias: array[lunes..domingo] of integer;

dia:integer;

begin

For dia:=lunes **to** domingo **do** lluvias[dia]:=sinvalor;

end.

¿Que pasaría si quisiéramos tener 4 variables del tipo arreglo, una para cada semana del mes?

program LluviaSemanal;

const

lunes=1;

domingo=7;

sinvalor=-1;

var

lluviaSemana1: array[lunes..domingo] of integer;

lluviaSemana2: array[lunes..domingo] of integer;

lluviaSemana3: array[lunes..domingo] of integer;

lluviaSemana4: array[lunes..domingo] of integer;

dia:integer;

begin

For dia:=lunes **to** domingo **do**

lluviaSemana1[dia]:=sinvalor;

.....

end.

Tipos

- Un tipo es un “molde” que define los posibles valores que se pueden tener y las operaciones
 - Tipos primitivos simples (integer, boolean...)
 - Tipos estructurados (arreglos...)
 - Tipos definidos por el usuario (Pila, Fila, ...)

program LluviaSemanal;

const

lunes=1;
domingo=7;
sinvalor=-1;

Type

lluvias= array[lunes..domingo] of integer;

var

lluviaSemana1, lluviaSemana2: lluvias;
dia:integer;

begin

For dia:= lunes to domingo do lluviaSemana1[dia]:=sinvalor;
end.

program LluviaSemanal;

const

lunes=1;

domingo=7;

sinvalor=-1;

Type

lluvias= array[lunes..domingo] of integer;

módulos (....)

var

lluviaSemana1, lluviaSemana2: lluvias;

dia:integer;

begin

For dia:= lunes **to** domingo **do** lluviaSemana1[dia]:=sinvalor;

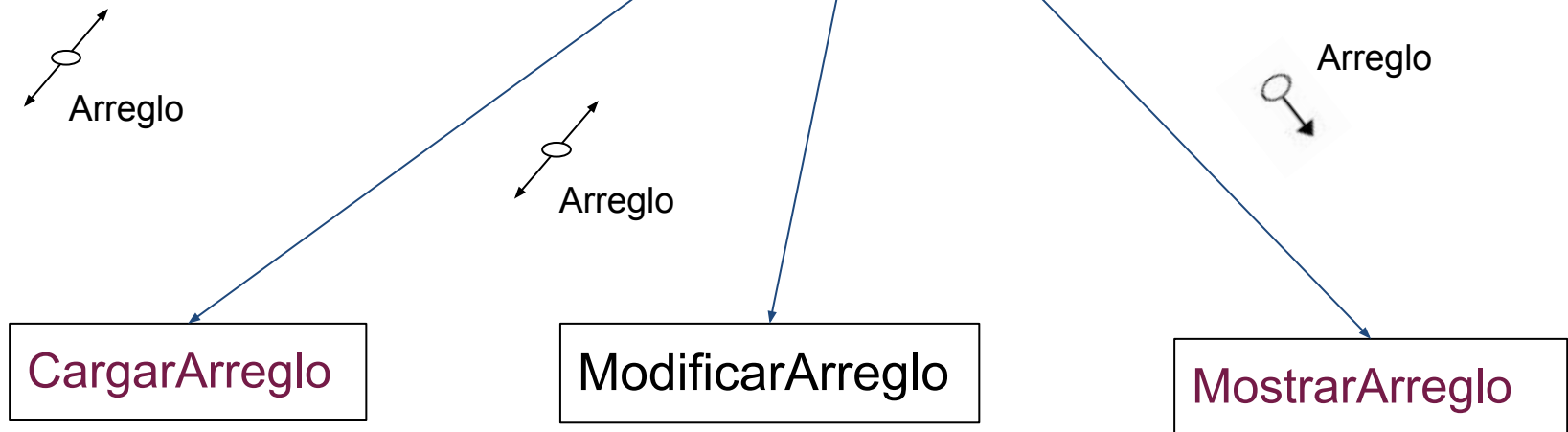
end.

Ventajas de Definir Tipos y constantes

- Hacen mas comprensible y autocontenido el código
- Disminuyen la posibilidad de cometer errores
- Brindan más velocidad y seguridad a la hora de incorporar cambios en el código
- Pascal No permite definir el tipo “array” como parámetro formal, por lo tanto hay que utilizar un tipo definido por el usuario
- Las constantes y los tipos se pueden usar en forma global porque su valor NO cambia durante la ejecución (por esa razón se usa =)
- Todas estas ventajas se potencian cuando más se reutiliza su definición, es decir, cuanto más variables o mas código utilizan constantes o tipos.

Realizar un programa que lea por teclado un arreglo de enteros de dimensión MaxArreglo; luego sume a cada celda una constante ValorAdicional.

IncrementarValorCeldasArreglos



Program IncrementarValorCeldasArreglos;

const

min=1;

maxArreglo= 4;

valorAdicional= 10;

type

ArregloNumeros= array[min..maxArreglo] of integer;

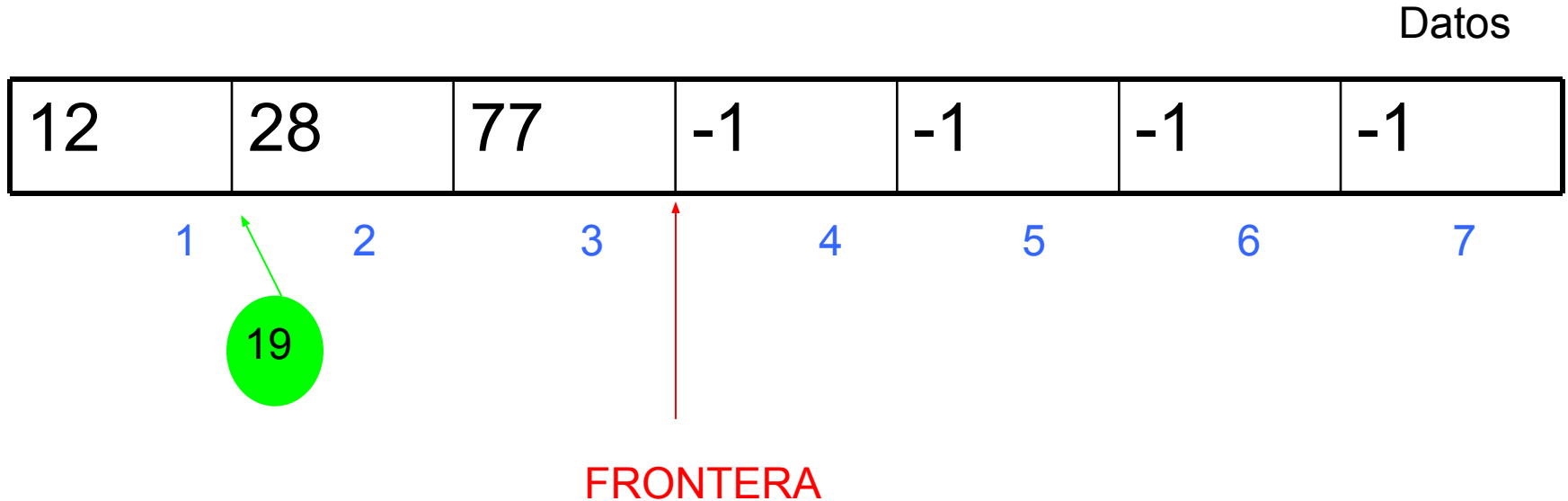
```
procedure CargarArreglo(var arrNum: arregloNumeros);
var
    indice: integer;
begin
    For indice:= min to MaxArreglo do begin
        write('Ingrese el valor del arreglo: ');
        readln(arrNum[indice]);
    end;
end;
```

```
procedure MostrarArreglo(arrNum: arregloNumeros);
var
    indice: integer;
begin
    For indice:= min to MaxArreglo do
        writeln('El valor del arreglo en la posición ', indice, 'es: ', arrNum[indice]);
    end;
```

```
procedure ModificarArreglo(var arrNum: arregloNumeros);  
var  
    indice: integer;  
begin  
    For indice:= min to MaxArreglo do  
        arrNum[indice]:= arrnum[indice] + valoradicional  
end;
```

```
var  
    arrNum: arregloNumeros;  
begin  
    CargarArreglo(arrNum);  
    MostrarArreglo(arrNum);  
    ModificarArreglo(arrNum);  
    writeln('El arreglo modificado quedo:');  
    MostrarArreglo(arrNum);  
end.
```

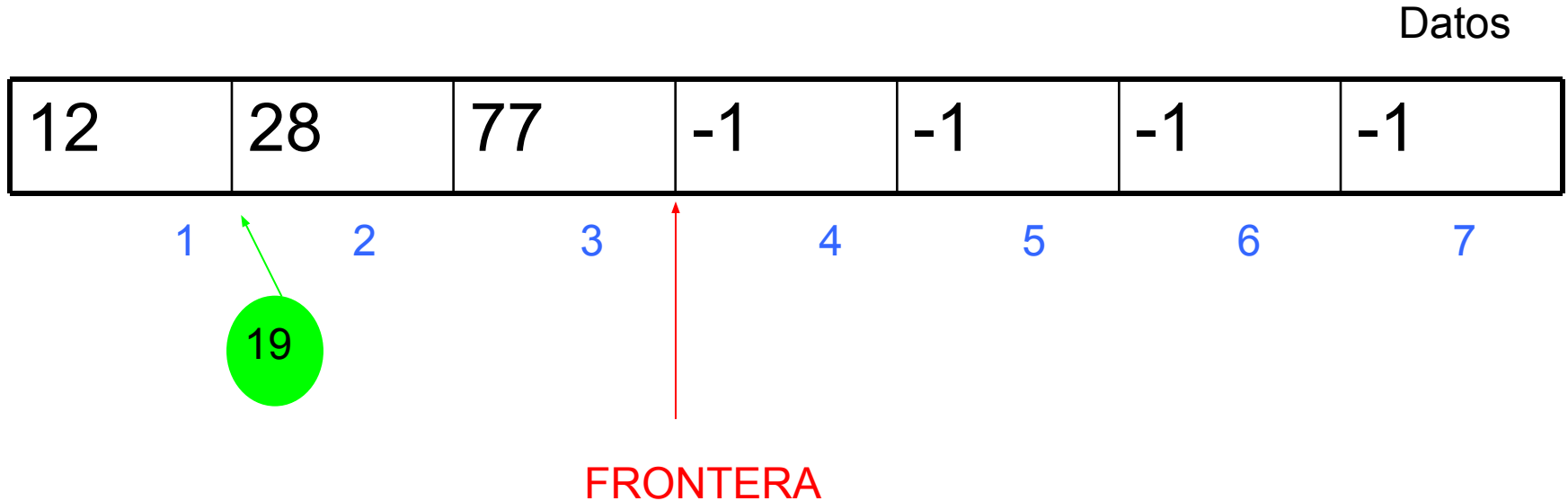
INSERCIÓN ORDENADA DE UN ELEMENTO



Problema:

Cargar en el arreglo DATOS el número 19 de tal forma que DATOS continúe ordenado de menor a mayor

INSERCIÓN ORDENADA DE UN ELEMENTO



Problema:

Cargar en el arreglo DATOS el número 19 de tal forma que DATOS continúe ordenado de menor a mayor

INSERCIÓN ORDENADA DE UN ELEMENTO

12	28	77	-1	-1	-1	-1
----	----	----	----	----	----	----



Datos

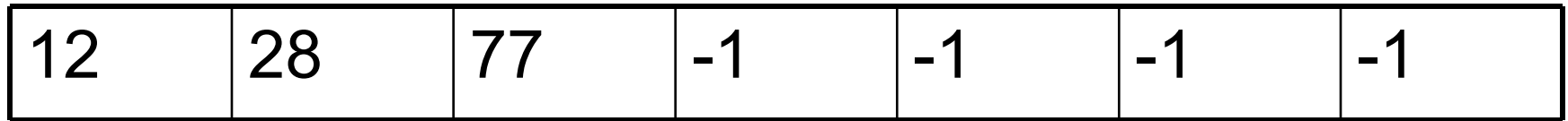
FRONTERA

Problema:

Cargar en el arreglo DATOS el número 19 de tal forma que DATOS continúe ordenado de menor a mayor

Se corre la frontera

Se deben desplazar los elementos comenzando en el final. Hasta cuando?



12	28	77	-1	-1	-1	-1
----	----	----	----	----	----	----

Datos

19

FRONTERA

Problema:

Cargar en el arreglo DATOS el número 19 de tal forma que DATOS continúe ordenado de menor a mayor

Se corre la Frontera

Se comienzan a desplazar los números hacia la derecha.

Se deben desplazar los elementos comenzando en el final. Hasta cuando?



12	28	77	77	-1	-1	-1
----	----	----	----	----	----	----

Datos

FRONTERA

Problema:

Cargar en el arreglo DATOS el número 19 de tal forma que DATOS continúe ordenado de menor a mayor

Se corre la frontera

Se comienzan a desplazar los numeros hacia la derecha

Se deben desplazar los elementos comenzando en el final. Hasta cuando?

12	28	28	77	-1	-1	-1
----	----	----	----	----	----	----

Datos

19

FRONTERA

Problema:

Cargar en el arreglo DATOS el número 19 de tal forma que DATOS continúe ordenado de menor a mayor

Se corre la frontera

Se comienzan a desplazar los numeros hacia la derecha

**Hasta cuando? Hasta encontrar su lugar (en este caso al mirar el Datos[1])
ó se acabaron los elementos**

Se deben desplazar los elementos comenzando en el final. Hasta cuando?



12	19	28	77	-1	-1	-1
----	----	----	----	----	----	----

Datos

FRONTERA

Problema:

Cargar en el arreglo DATOS el número 19 de tal forma que DATOS continúe ordenado de menor a mayor

- Se corre la frontera
- Se comienzan a desplazar los números hacia la derecha
- . Hasta cuando? Hasta encontrar su lugar (en este caso al mirar el Datos[1])
- Se inserta el valor.

BORRADO DE UN ELEMENTO

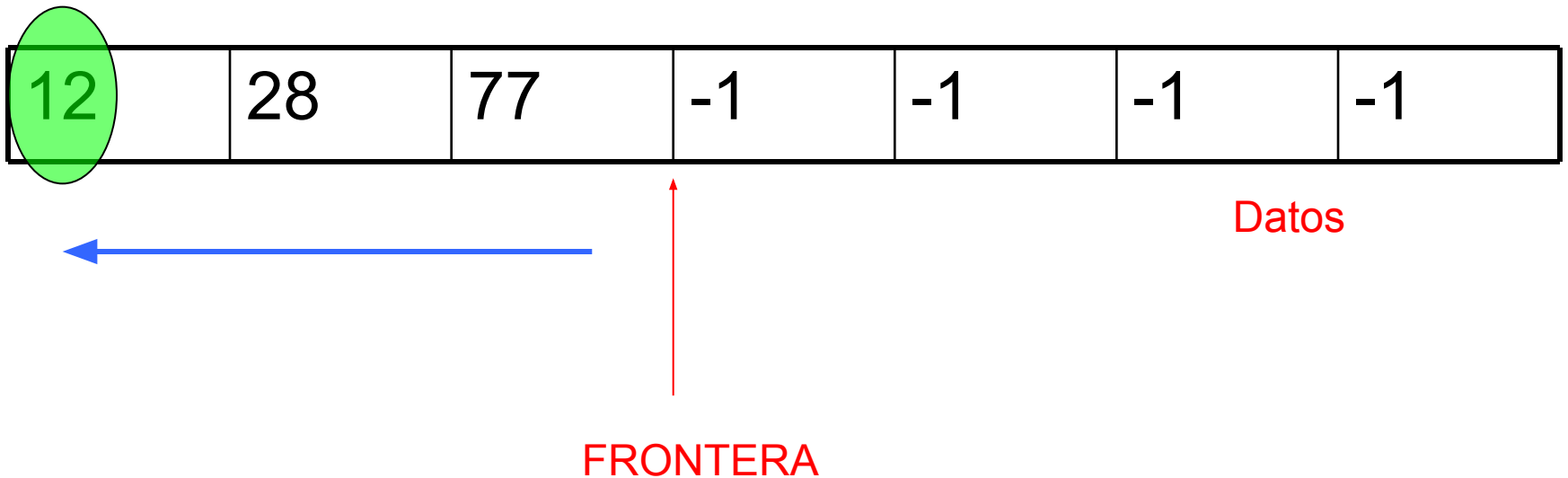
12	28	77	-1	-1	-1	-1
----	----	----	----	----	----	----

Datos

FRONTERA

Problema:

Eliminar del arreglo **DATOS** el número 12 de tal forma que DATOS continúe ordenado de menor a mayor



Problema:

Eliminar del arreglo **DATOS** el número 12 de tal forma que DATOS continúe ordenado de menor a mayor

Se busca el elemento

Se desplazan los elementos hacia la izquierda

Hasta cuando? Hasta el elemento a borrar

28	28	77	-1	-1	-1	-1
----	----	----	----	----	----	----

Datos



FRONTERA

Problema:

Eliminar del arreglo **DATOS** el número 12 de tal forma que DATOS continúe ordenado de menor a mayor

Se busca el elemento

Se desplazan los elementos hacia la izquierda

Hasta cuando? Hasta el elemento a borrar

28	77	77	-1	-1	-1	-1
----	----	----	----	----	----	----

Datos



FRONTERA

Problema:

Eliminar del arreglo **DATOS** el número 12 de tal forma que DATOS continúe ordenado de menor a mayor

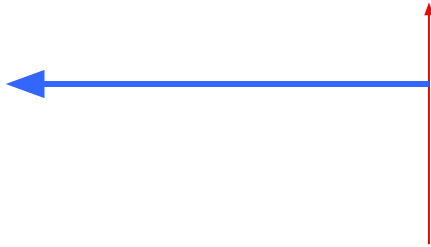
Se busca el elemento

Se desplazan los elementos hacia la izquierda

Hasta cuando? Hasta el elemento a borrar

28	77	77	-1	-1	-1	-1
----	----	----	----	----	----	----

Datos



FRONTERA

Problema:

Eliminar del arreglo **DATOS** el número 12 de tal forma que DATOS continúe ordenado de menor a mayor

Se busca el elemento

Se desplazan los elementos hacia la izquierda

Hasta cuando? Hasta el ultimo elemento

Se disminuye en uno el valor de la FRONTERA

- Realice una función que recibe a un entero y un arreglo de enteros como parámetro y devuelve la posición donde se encuentra el elemento dentro del arreglo, sino está devuelve -1.

Program EjemploArreglos;

const

min=1;

maxArreglo=4;

type

ArregloNumeros= array[min..maxArreglo] of integer;

```
Function PosiciónDatoEnArreglo(dato:integer; arreNum:  
ArregloNumeros): integer;
```

```
var
```

```
    indice: integer;
```

```
begin
```

```
    indice:=1;
```

```
    while (indice <= maxArreglo) and (arreNum[indice] <> dato) do
```

```
        indice:= indice +1;
```

```
    if indice > maxArreglo
```

```
        then
```

```
            PosiciónDatoEnArreglo:= -1
```

```
        else
```

```
            PosiciónDatoEnArreglo:= indice;
```

```
end;
```

Métodos de Búsqueda

- Procesos que involucren recorrer una lista de elementos con el fin de encontrar alguno.
- Búsqueda lineal o secuencial
- Búsqueda binaria

Búsqueda Secuencial o lineal

No se conoce sobre la manera en que la estructura está organizada.

Se comienza desde el inicio de la estructura comparando el elemento que se busca con cada elemento de la lista de elementos hasta cuando se encuentra o se termina la lista.

Se quiere buscar el 4

96	16	77	99	5	12	1	88	9	0	4	98	7
----	----	----	----	---	----	---	----	---	---	---	----	---

Se quiere buscar el 4

96	16	77	99	5	12	1	88	9	0	4	98	7
----	----	----	----	---	----	---	----	---	---	---	----	---

Y si está ordenado?

0	1	4	5	7	9	12	16	77	88	96	98	99
---	---	---	---	---	---	----	----	----	----	----	----	----

Y si está ordenado?

0	1	4	5	7	9	12	16	77	88	96	98	99
---	---	---	---	---	---	----	----	----	----	----	----	----

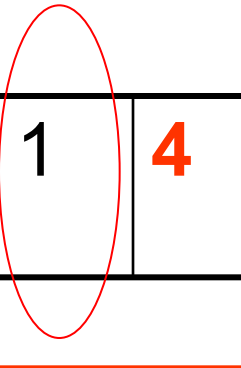


Y si está ordenado?

0	1	4	5	7	9	12	16	77	88	96	98	99
---	---	---	---	---	---	----	----	----	----	----	----	----



Y si está ordenado?



0	1	4	5	7	9	12	16	77	88	96	98	99
---	---	---	---	---	---	----	----	----	----	----	----	----

Y si está ordenado?

0	1	4	5	7	9	12	16	77	88	96	98	99
---	---	---	---	---	---	----	----	----	----	----	----	----

—

Búsqueda Binaria

- Se trabaja sobre una estructuras ordenadas de elementos.
- Estrategia divide y conquista:
 - Sobre una estructura compara el valor buscado con el valor de la mitad de la estructura.
 - Si es el mismo, finaliza, sino toma la mitad superior o inferior de la estructura (según el valor buscado)
 - Para esa parte, se hace el mismo procedimiento y así sucesivamente hasta encontrar el valor o no poder dividir (el elemento no está).