TEMAS DE INTRO I

- Pilas (datos/estructuras de control)
- Filas (datos/estructuras de control)
- Modularización y Parámetros
- Variables
- Método de Desarrollo
- Funciones
- Arreglos
- Matrices / Trabajo Especial
- Revisión general

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

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). Se definen 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 Letras (capacidad 4)

´a´	´b´	´v	´ñ´
1	2	3	4

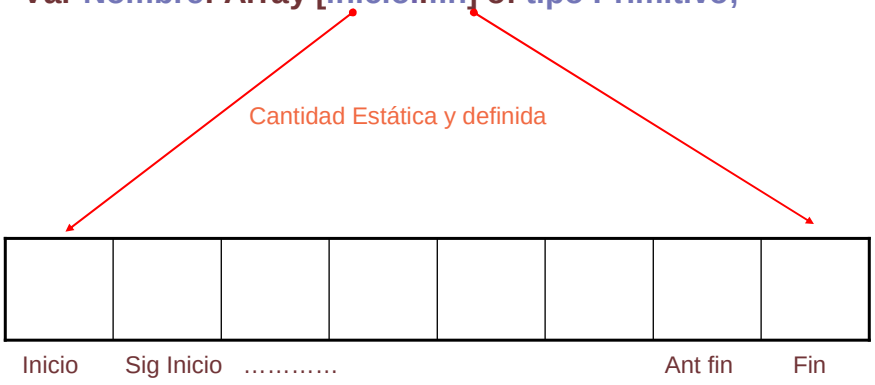
Posición ó Índice

Es un Tipo de Datos

ESTRUCTURADO
HOMOGÉNEO
ESTÁTICO
INDEXADO

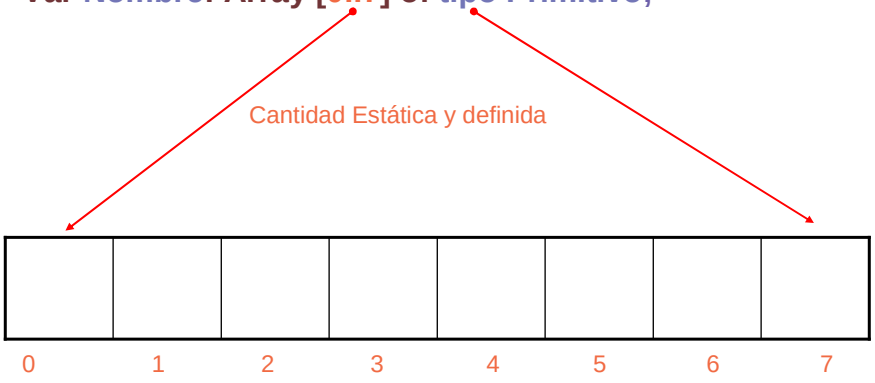
Arreglos en Pascal

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



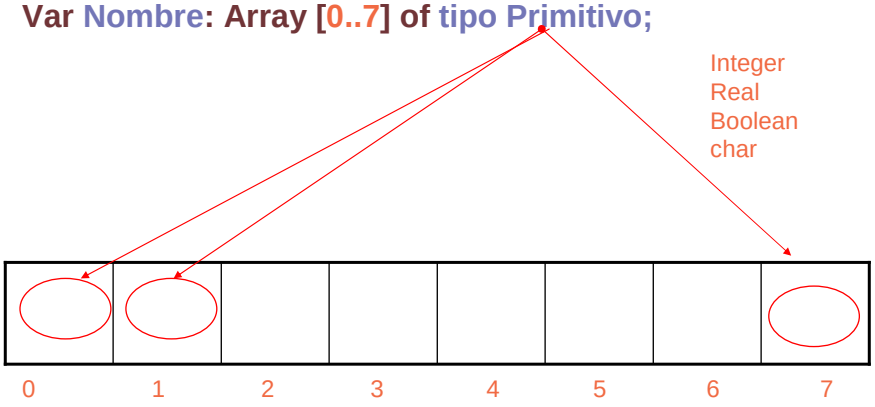
Arreglos en Pascal

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



Arreglos en Pascal

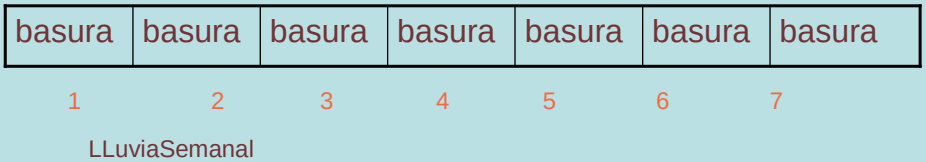
Var Nombre: Array [0..7] of tipo Primitivo;



Arreglos en Pascal

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

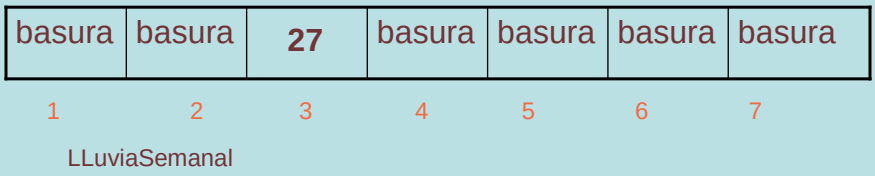
MEMORIA DE LA MÁQUINA



Arreglos en Pascal

```
Var LluviaSemanal: Array [1..7] of Integer
Begin
  LLuviaSemana[3]:=27;
```

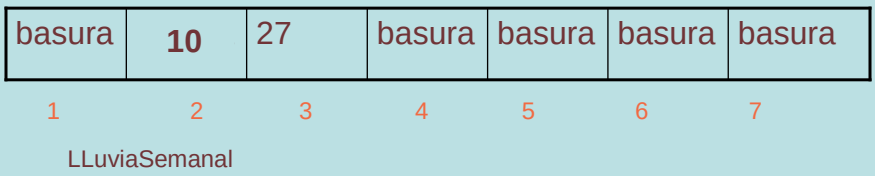
MEMORIA DE LA MÁQUINA



Arreglos en Pascal

```
Var LluviaSemanal: Array [1..7] of Integer
Begin
  LLuviaSemana[3]:=27;
  LLuviaSemana[2]:=10;
```

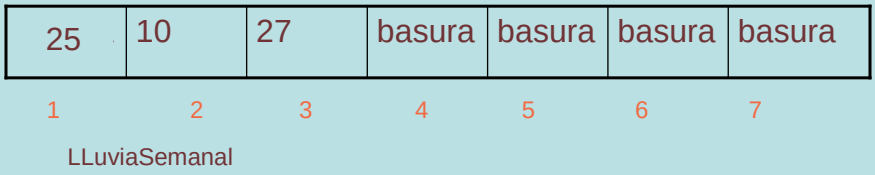
MEMORIA DE LA MÁQUINA



Arreglos en Pascal

```
Var LluviaSemanal: Array [1..7] of Integer
Begin
  LLuviaSemana[3]:=27;
  LLuviaSemana[2]:=10;
  LLuviaSemana[1]:=15 +  LLuviaSemana[2];
```

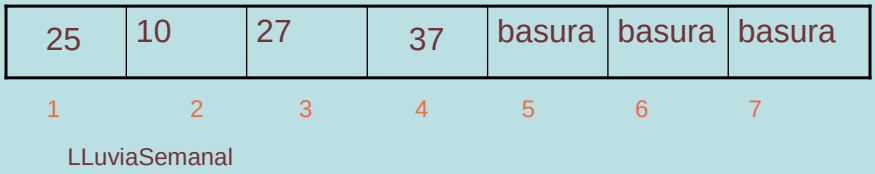
MEMORIA DE LA MÁQUINA



Arreglos en Pascal

```
Var LluviaSemanal: Array [1..7] of Integer
Begin
  LLuviaSemana[3]:=27;
  LLuviaSemana[2]:=10;
  LLuviaSemana[1]:=15 +  LLuviaSemana[2];
  LLuviaSemana[4]:=  LLuviaSemana[2] +  LLuviaSemana[3];
```

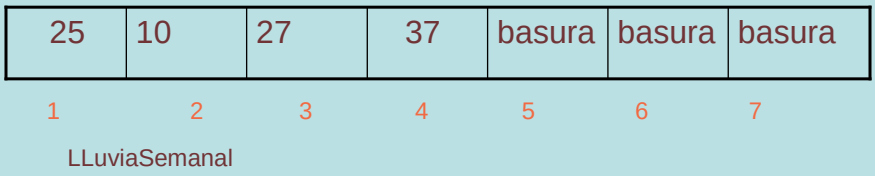
MEMORIA DE LA MÁQUINA



Arreglos en Pascal

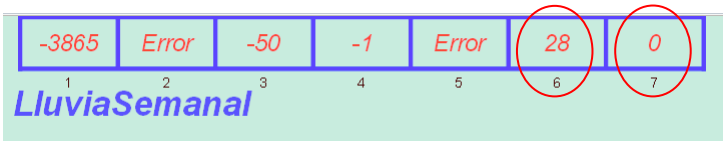

```
Var LluviaSemanal: Array [1..7] of Integer
Begin
  LLuviaSemana[3]:=27;
  LLuviaSemana[2]:=10;
  LLuviaSemana[1]:=15 +  LLuviaSemana[2];
  LLuviaSemana[4]:=  LLuviaSemana[2] +  LLuviaSemana[3];
  Writeln(LLuviaSemana[1]);
```

MEMORIA DE LA MÁQUINA



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"



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
              LluviaSemana[día] := -1;
              día:= día +1
        end;
.........
```

Índice

- Estructura de control FOR

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

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

Volviendo al control de llenado con un valor discernible

```
FOR dia :=1 to 7   DO   LluviaSemana[dia] := -1;
……..……..
LluviaSemana[2] := 6;
LluviaSemana[3] := 7;
LluviaSemana[5] := 17;
LluviaSemana[6] := 9;

For  dia := 1 to 7 do
   if lluviaSemana[dia] <> -1 then
       writeln(LluviaSemana[dia]);
……..……..
```

Sólo se utilizan las celdas cuyo valor es <> -1

LLuviaSemanal

-1	6	7	-1	17	9	-1
1	2	3	4	5	6	7

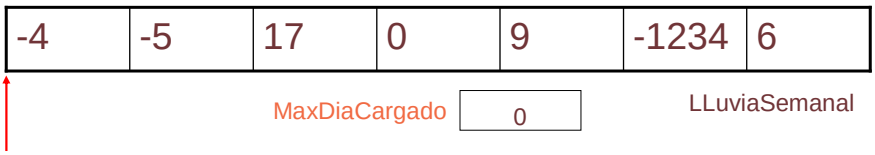
Control de llenado
posición límite o Frontera

4	5	17	0	9	-1234	6

LLuviaSemanal

FRONTERA

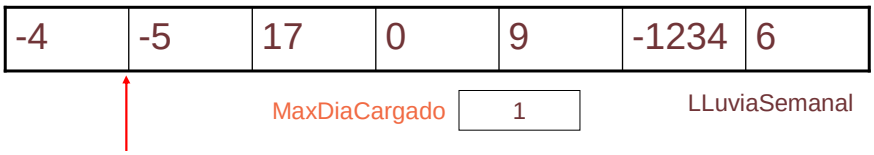
1- La forma de carga debe ser consecutivas sin espacios “vacíos”
2- Definir una variable que marcará la FRONTERA
dividiendo la FRONTERA entre las celdas utilizadas y las no utilizadas.
3- La variable FRONTERA debe ser utilizada tanto para agregar valores como para consultar los ya agregados.
4- Se simula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



MaxDiaCargado 0

LLuviaSemanal

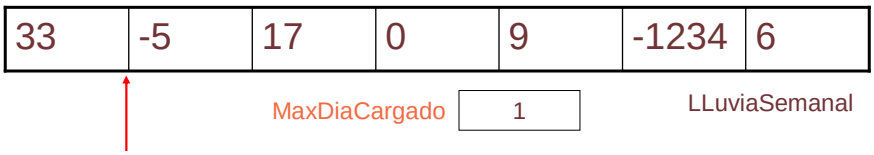
```
Program    EjemploArregloFrontera;
{ Este porción de programa muestra un ejemplo de manejo de arreglos con Frontera}

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.
```



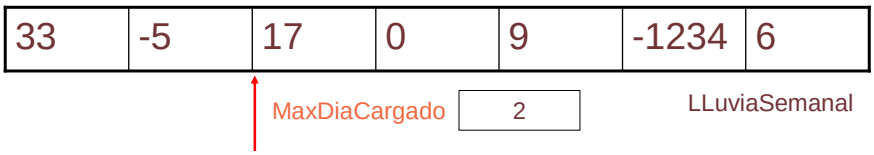
MaxDiaCargado 1

LLuviaSemanal

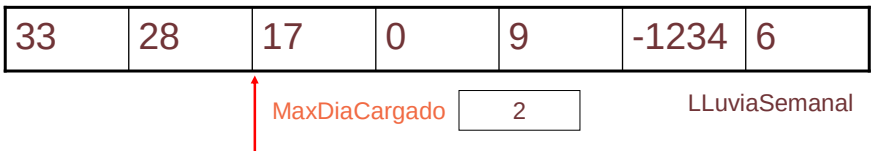
```
Program    EjemploArregloFrontera;
{ Este porción de programa muestra un ejemplo de manejo de arreglos con Frontera}

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



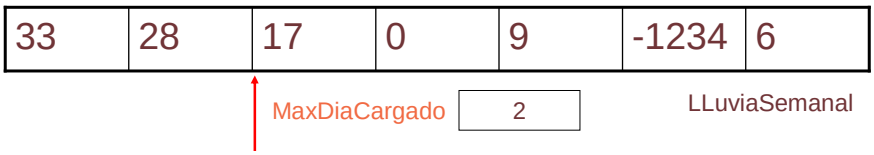

```
Program    EjemploArregloFrontera;
{ Este porción de programa muestra un ejemplo de manejo de arreglos con Frontera}

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



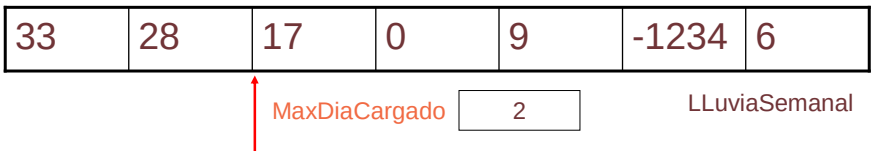

```
Program    EjemploArregloFrontera;
{ Este porción de programa muestra un ejemplo de manejo de arreglos con Frontera}

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.
```

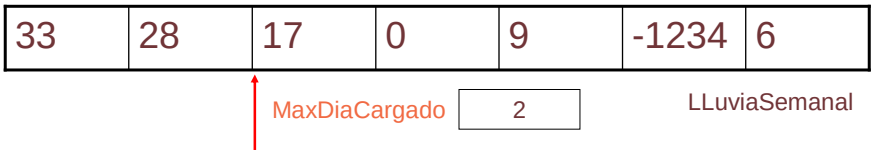



```
Program    EjemploArregloFrontera;
{ Este porción de programa muestra un ejemplo de manejo de arreglos con Frontera}

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.
```



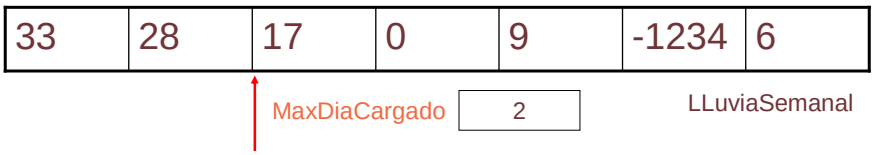

```
Program    EjemploArregloFrontera;
{ Este porción de programa muestra un ejemplo de manejo de arreglos con Frontera}

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

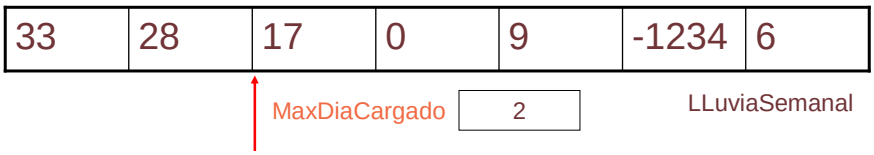



```
Program    EjemploArregloFrontera;
{ Este porción de programa muestra un ejemplo de manejo de arreglos con Frontera}

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.
```

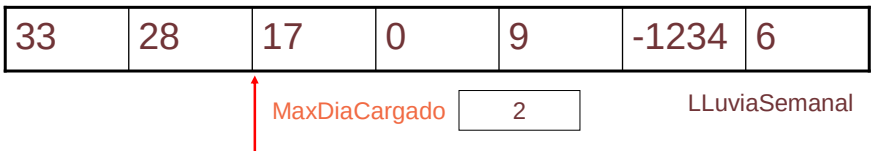
33




```
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  28
                                   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;
     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  28
                                   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.
```



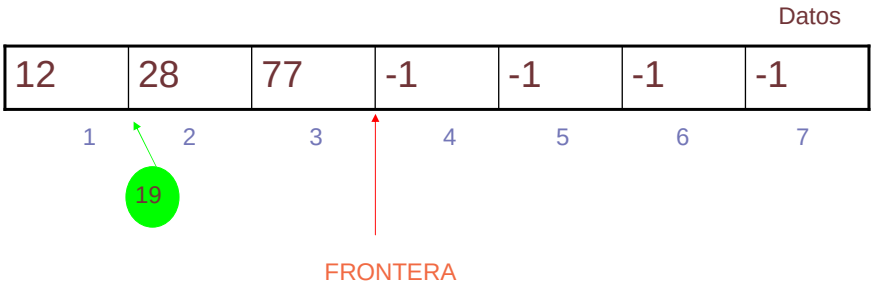

```
Program    EjemploArregloFrontera;
{ Este porción de programa muestra un ejemplo de manejo de arreglos con Frontera}

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  28
                                   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.
```

Ejercicio

- Hacer un procedimiento/función (cuál es mas adecuado?) que devuelva la posición de un determinado número en el arreglo NUMEROS. Si no existe, devuelve un -1

Nota: Considerar el manejo de una variable FRONTERA que marca hasta donde tiene cargados elementos el arreglo

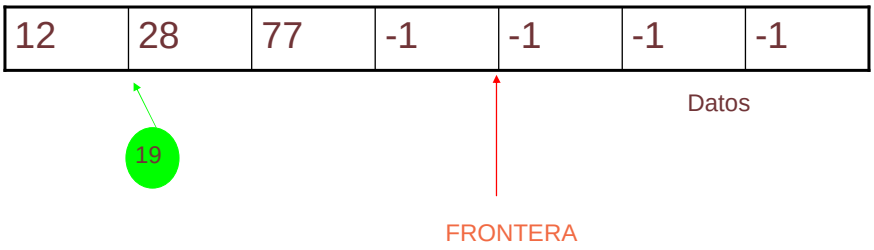
INSERCION DE UN ELEMENTO



Problema:

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

INSERCION DE UN ELEMENTO

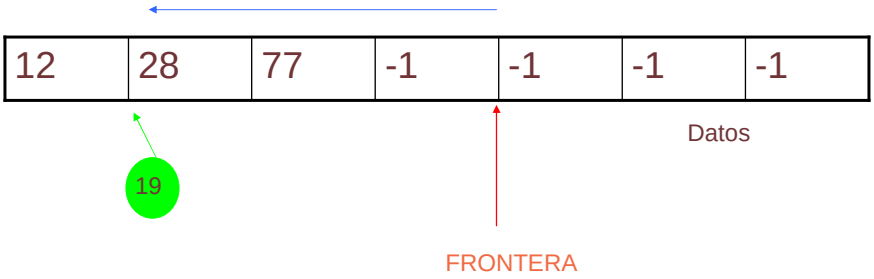


Problema:

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?

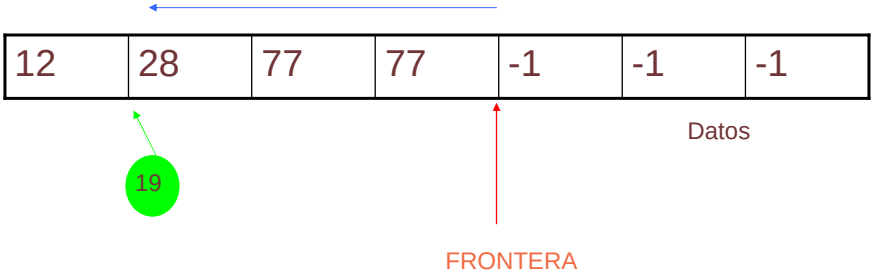


Problema:

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?

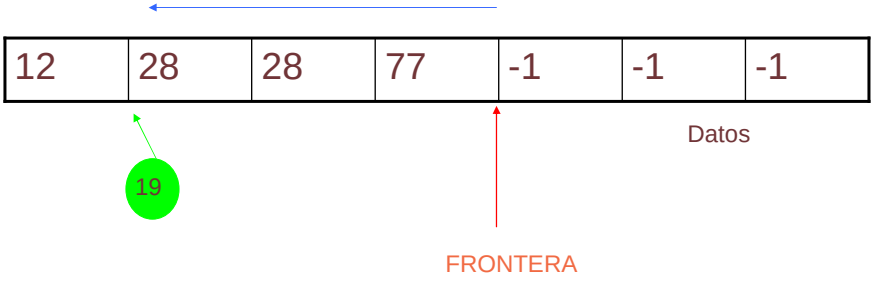


Problema:

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?

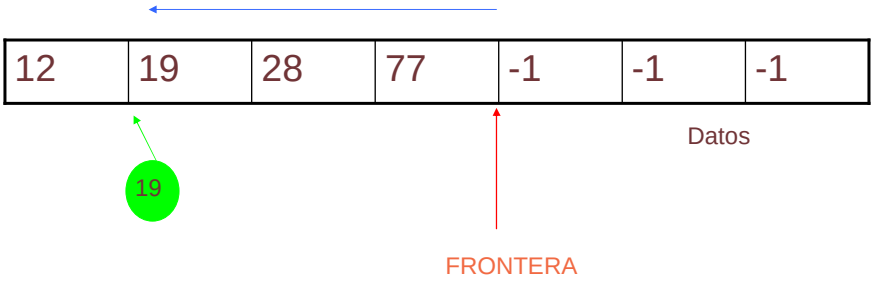


Problema:

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?

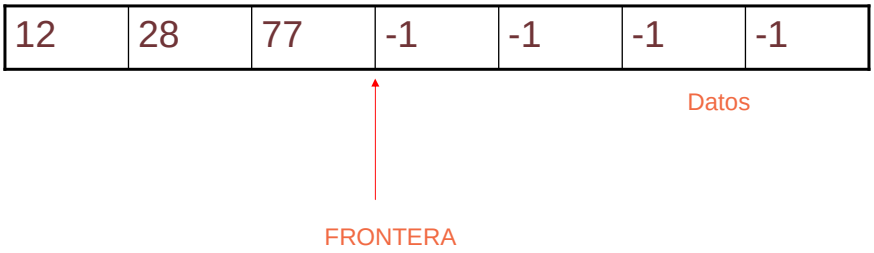


Problema:

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

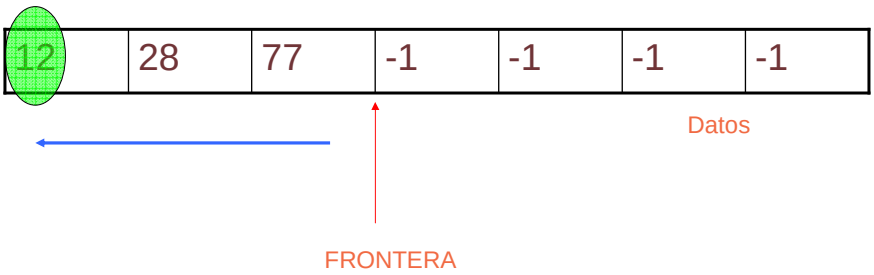
- Se corre la frontera
- Se comienzan a desplazar los números hacia la derecha
. Hasta cuando? Hasta encontrar su lugar (en este caso al mirar el Datos[1])
- Se inserta el valor.

BORRADO DE UN ELEMENTO



Problema:

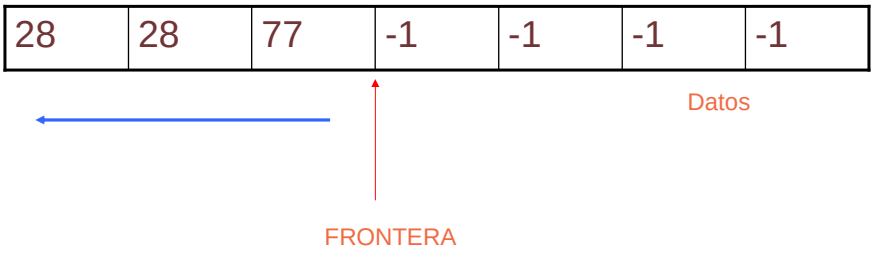
Eliminar del arreglo DATOS el número 12 de tal forma que DATOS continúe ordenado de menor a mayor



Problema:

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

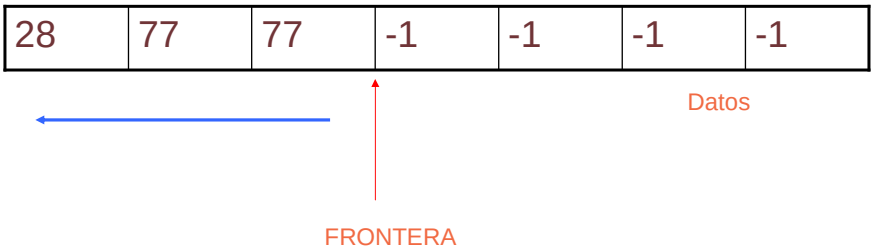
Se busca el elemento
Se desplazan los elementos hacia la izquierda
Hasta cuando? Hasta el ultimo elemento



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



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

28	77	77	-1	-1	-1	-1

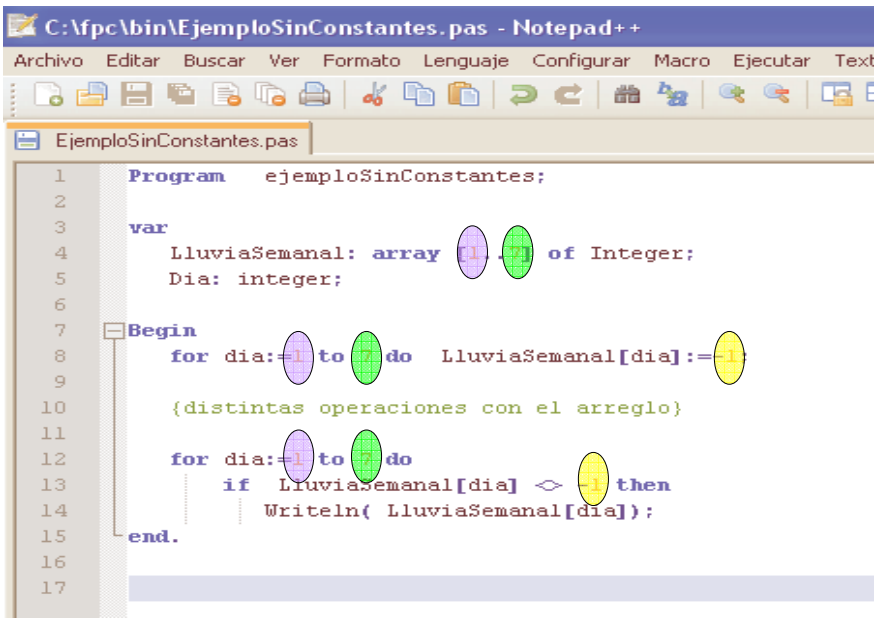
Datos

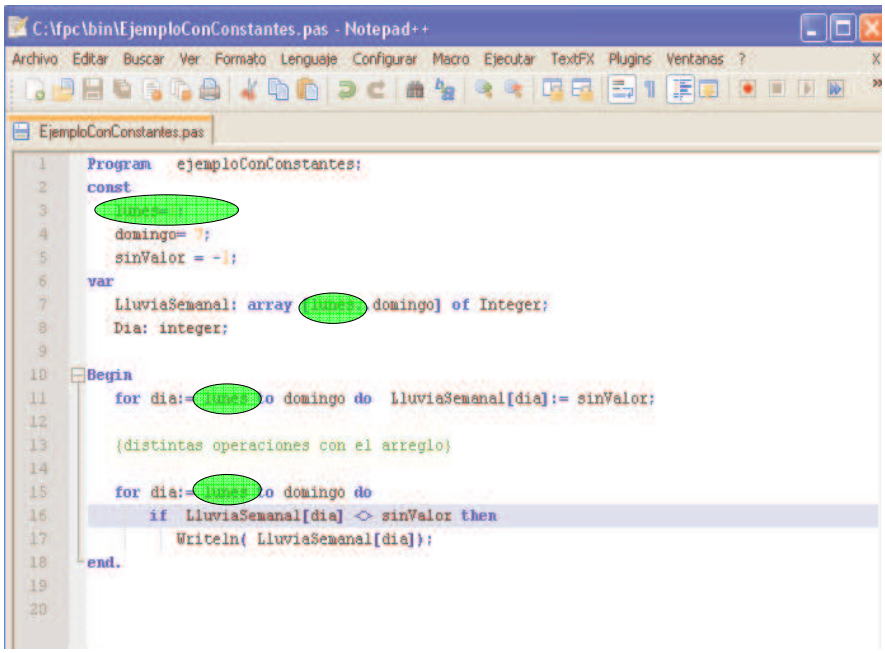
FRONTERA

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





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 LEGILIBILIDAD 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.
```

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;
var
   lluvias: array[lunes..domingo] of integer;
   dia:integer;
begin
   For dia:=1 to domingo do lluvias[dia]:=sinvalor;
end.
```

¿Que pasaría si quisiéramos tener 4 variables de lluvia, 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:=1 to domingo do lluviaSemana1[dia]:=sinvalor;
……..
end.
```

```
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;
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.