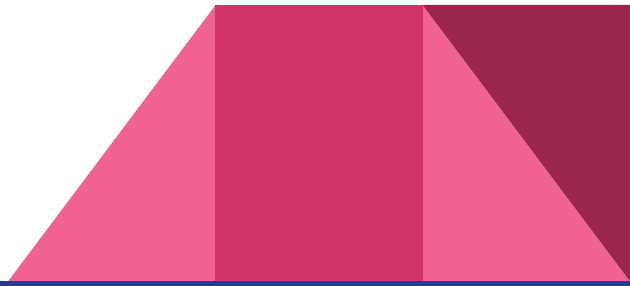
Introducción a la Programación I

- **La estructura de la cursada**
- El sitio de la materia
- Términos y conceptos

Días y horarios

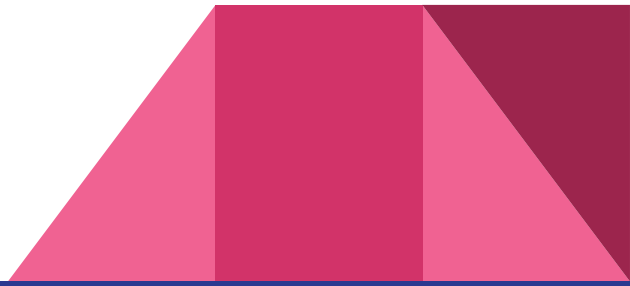
- Teorías
  - Martes
- Prácticas
  - Miercoles

Estar atentos a mensajes via moodle enviados desde la cátedra : introprog@alumnos.exa.unicen.edu.ar



Cursada

- Un tema / práctico por semana.
- Cada semana se comienza con más teoría y se termina con práctica.
- Se considera que los estudiantes no tienen ningún conocimiento previo de programación
- Programación en Pascal



Cursada, final, notas, evaluaciones

Evaluaciones dentro de la cursada:

- Parcial / Recuperatorio / Prefinal
- Trabajo (1, 2 ó 3)

Promoción

Final

Copias

Introducción a la Programación I

- ✓ La estructura de la cursada
- ✓ **El sitio de la materia y la Página de la Facultad**
- ✓ Términos y conceptos

Introducción a la Programación I

- ✓ La estructura de la cursada
- ✓ El sitio de la materia
- ✓ Términos y conceptos

Términos y conceptos

Programa, Datos, Control, Lenguaje, Procesador, Proceso

TORTA DE PERAS Y NUECES

(https://cocinerosargentinos.com/pasteleria/torta-de-peras-y-nueces)

INGREDIENTES

- 200 g de harina.
- 2 cditas de polvo para hornear.
- 200 g de azúcar rubia.
- 100 cc de aceite.-
- 2 huevos.
- Esencia de vainilla.
- 3 peras.
- Un limón.
- 100 g de almendras.

🍐 TORTA DE PERAS Y NUECES 🍐 ¡súper fácil de hacer en tu casa! seguí el paso a paso a continuación

1. Batir a mano los huevos con el azúcar. **2.** Agregar el aceite, la esencia y ralladura de un limón

3. Pelar y cortar las peras en cuadrados chicos y volcar en la preparación.**4.** Incorporar la harina con el polvo y las almendras. **5.** Enmantecar y espolvorear con azúcar el molde.

6. Volcar la preparación y poner las nueces partidas por arriba **7.** Llevar a horno mínimo (150°C) durante 45 minutos.

Similitudes

Receta	Programa
Ingredientes	Datos
Pasos	Control, Secuencias
Chef	Procesador
Idioma	Lenguaje de programación
Una elaboración	Proceso

PILAS

Receta	Programa
Ingredientes	Datos
Pasos	Control, Secuencias
Chef	Procesador
Idioma	Lenguaje de programación
Una elaboración	Proceso

SECUENCIAS;
SELECCIÓN;
ITERACIÓN

PASCAL

Pilas

Una Pila es una colección de elementos del mismo tipo organizada como una superposición ordenada de los mismos.
Por ejemplo Pila de platos, Pila de cartas, Pila de camisas, Pila de libros

TOPE DE LA PILA

Los elementos se recuperan en el orden inverso al que fueron almacenados.
El último en entrar (TOPE) es el primer en salir.

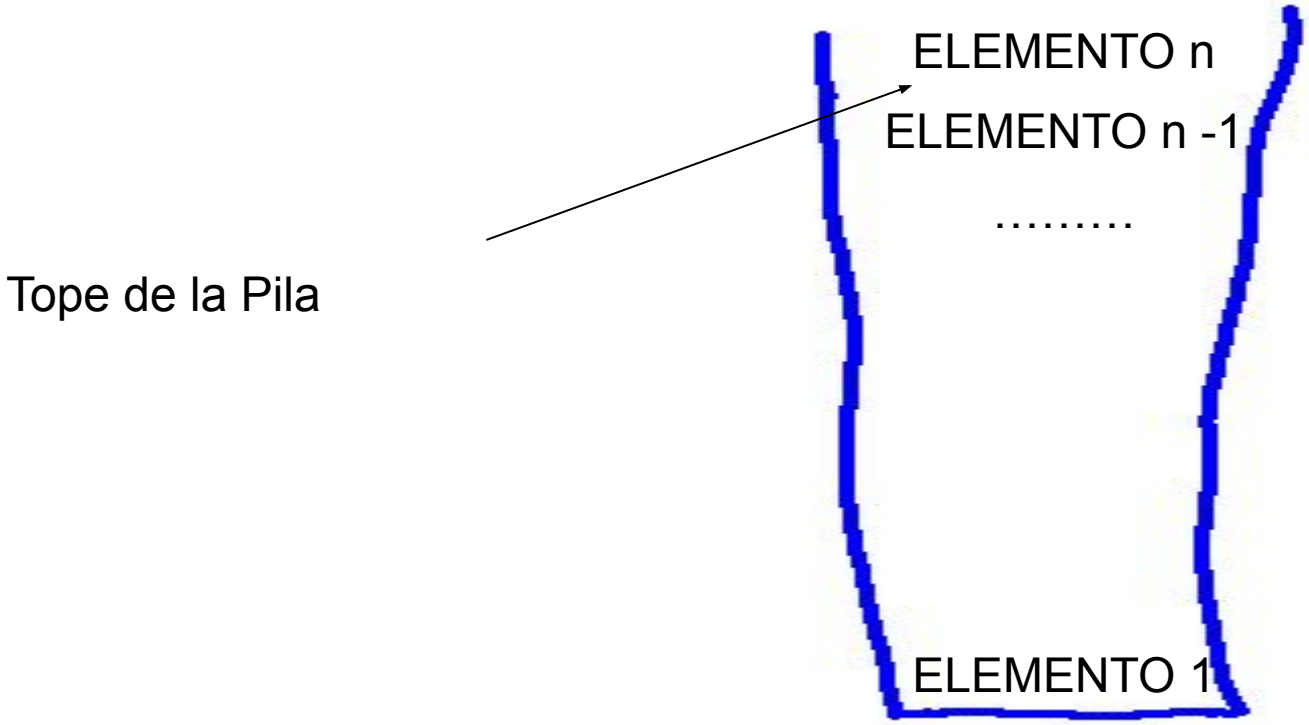
Pilas

- **Que se puede hacer sobre una pila?**

Apilar
Desapilar
VerTope
…..

Pilas

- **Representación Gráfica**

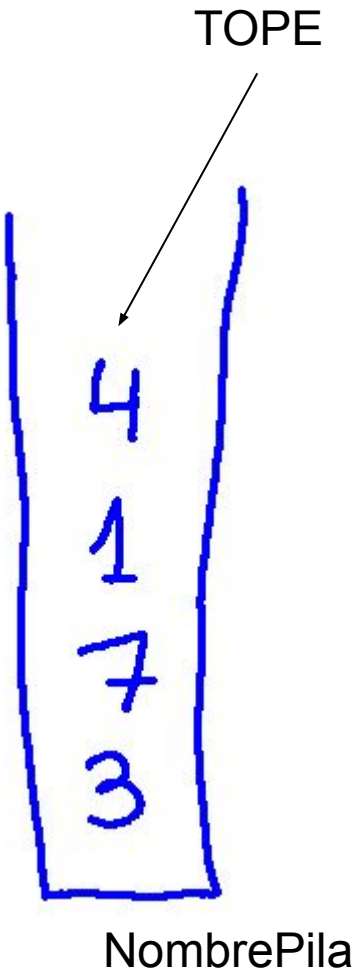


Pilas en Pascal

En Pascal vamos a trabajar con Pilas de Enteros.

Cada pila tendrá un nombre.

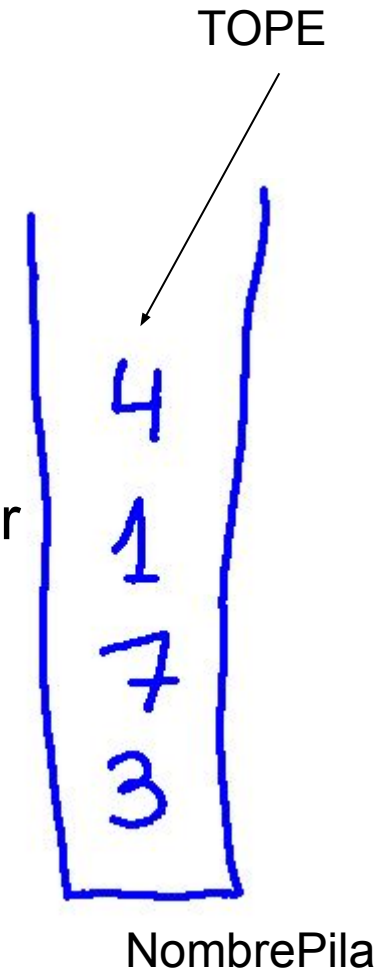
Sólo se ve el TOPE, no se sabe cuantos elementos hay



Pilas en Pascal

Se podrán hacer las siguientes operaciones :

- ReadPila()
- InicPila ()
- Tope()
- Apilar ()
- Desapilar() el Elemento que se desapila debe ir a otra Pila
- PilaVacia()
- WritePila()



PrimerPrograma.pas

```
1  program PrimerPrograma;
2
3  {Este programa permite cargar una pila por teclado,
4  la pasa a otra pila y la imprime}
5
6  {$INCLUDE /IntroProg/Estructu}
7
8  var origen, destino: pila;
9
10 begin
11     readpila(origen);
12     inicpila(destino, '');
13     while not pilavacia(origen) do begin
14         apilar (destino, desapilar(origen));
15     end;
16     writepila(destino);
17 end.
```

NOMBRE

PrimerPrograma.pas

```
1  program PrimerPrograma;
2
3  {Este programa permite cargar una pila por teclado,
4  la pasa a otra pila y la imprime}
5
6  {$INCLUDE /IntroProg/Estructu}
7
8  var origen, destino: pila;
9
10 begin
11     readpila(origen);
12     inicpila(destino, '');
13     while not pilavacia(origen) do begin
14         apilar (destino, desapilar(origen));
15     end;
16     writepila(destino);
17 end.
```

PrimerPrograma.pas

```
1  program PrimerPrograma;
2
3  {Este programa permite cargar una pila por teclado,
4  la pasa a otra pila y la imprime}
5
6  {$INCLUDE /IntroProg/Estructu}
7
8  var origen, destino: pila;
9
10 begin
11     readpila(origen);
12     inicpila(destino, '');
13     while not pilavacia(origen) do begin
14         apilar (destino, desapilar(origen));
15     end;
16     writepila(destino);
17 end.
```

Comentarios

PrimerPrograma.pas

```
1  program PrimerPrograma;
2
3  {Este programa permite cargar una pila por teclado,
4  la pasa a otra pila y la imprime}
5
6  {$INCLUDE /IntroProg/Estructu}
7
8  var origen, destino: pila;
9
10 begin
11     readpila(origen);
12     inicpila(destino, '');
13     while not pilavacia(origen) do begin
14         apilar (destino, desapilar(origen));
15     end;
16     writepila(destino);
17 end.
```

Estructu es una Unidad que permite trabajar con pilas en Pascal

PrimerPrograma.pas

```
1  program PrimerPrograma;
2
3  {Este programa permite cargar una pila por teclado,
4  la pasa a otra pila y la imprime}
5
6  {$INCLUDE /IntroProg/Estructu}
7
8  var origen, destino: pila;
9
10 begin
11     readpila(origen);
12     inicpila(destino, '');
13     while not pilavacia(origen) do begin
14         apilar (destino, desapilar(origen));
15     end;
16     writepila(destino);
17 end.
```

Se definen los datos con los que va a trabajar el programa.

PrimerPrograma.pas

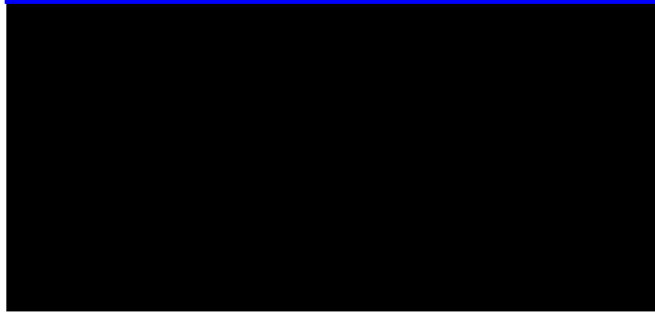
```
1  program PrimerPrograma;
2
3  {Este programa permite cargar una pila por teclado,
4  la pasa a otra pila y la imprime}
5
6  {$INCLUDE /IntroProg/Estructu}
7
8  var origen, destino: pila;
9
10 begin
11     readpila(origen);
12     inicpila(destino, '');
13     while not pilavacia(origen) do begin
14           apilar (destino, desapilar(origen));
15     end;
16     writepila(destino);
17 end.
```

Se escriben las instrucciones que trabajarán sobre los datos para lograr el objetivo del programa.

PrimerPrograma.pas

```
1  program PrimerPrograma;
2
3  {Este programa permite cargar una pila por teclado,
4  la pasa a otra pila y la imprime}
5
6  {$INCLUDE /IntroProg/Estructu}
7
8  var origen, destino: pila;
9
10 begin
11     readpila(origen);
12     inicpila(destino, '');
13     while not pilavacia(origen) do begin
14           apilar (destino, desapilar(origen));
15     end;
16     writepila(destino);
17 end.
```

C:\fpc\bin\Inicial



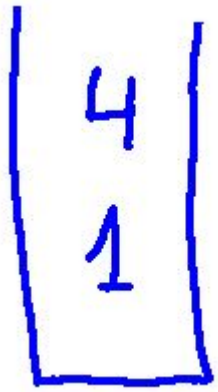
```
1  Program Inicial;
2  {Este Programa es un  ejemplo sencillo de manejo
3  de Pilas en Pascal:
4  Leer los datos de una Pila por pantalla e imprimirlos }
5
6  {$INCLUDE /IntroProg/Estructu}
7  var
8    Origen: pila;
9
10 Begin
11   ReadPila(Origen);
12   WritePila(Origen);
13 end.
```

Estado de las Pilas durante ejecución:

Origen

```
C:\fpc\bin\Inicial

Ingresar elementos a la Pila:
<Base><...><Tope>
     1      4
```

```
1  Program Inicial;
2  {Este Programa es un  ejemplo sencillo de manejo
3  de Pilas en Pascal:
4  Leer los datos de una Pila por pantalla e imprimirlos }
5
6  {$INCLUDE /IntroProg/Estructu}
7  var
8    Origen: pila;
9
10 Begin
11   ReadPila(Origen);
12   WritePila(Origen);
13 end.
```

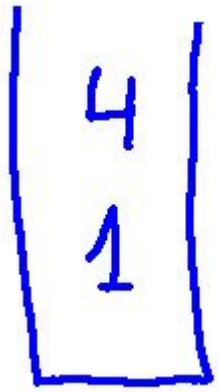


Estado de las Pilas durante ejecución:

Origen 1 4

```
1  Program Inicial;
2  {Este Programa es un  ejemplo sencillo de manejo
3  de Pilas en Pascal:
4  Leer los datos de una Pila por pantalla e imprimirlos }
5
6  {$INCLUDE /IntroProg/Estructu}
7
8  var
9    Origen: pila;
10  Begin
11   ReadPila(Origen);
12   WritePila(Origen);
13 end.
```

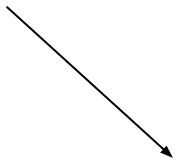
```
C:\fpc\bin\Inicial

Ingresar elementos a la Pila:
<Base><...><Tope>

      1      4

<Base>   1    4   <Tope>
```



Estado de las Pilas durante ejecución:

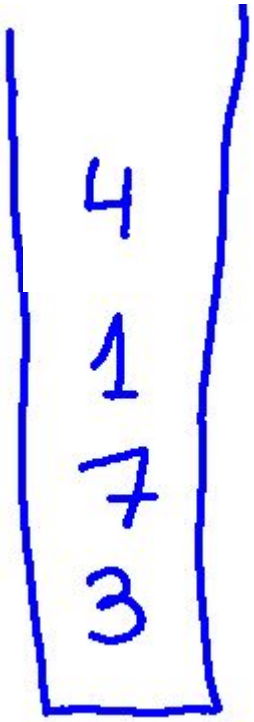
Origen	1
	4

```
Program Inicial;
{Este Programa es un  ejemplo sencillo de manejo
de Pilas en Pascal:
Leer los datos de una Pila por pantalla e imprimirlos }

{$INCLUDE /IntroProg/Estructu}
var
  Origen: pila;

Begin
  ReadPila(Origen);
  WritePila(Origen);
end.
```



Sentencias Secuenciales: se ejecutan una después de la otra

Operaciones válidas con Pilas



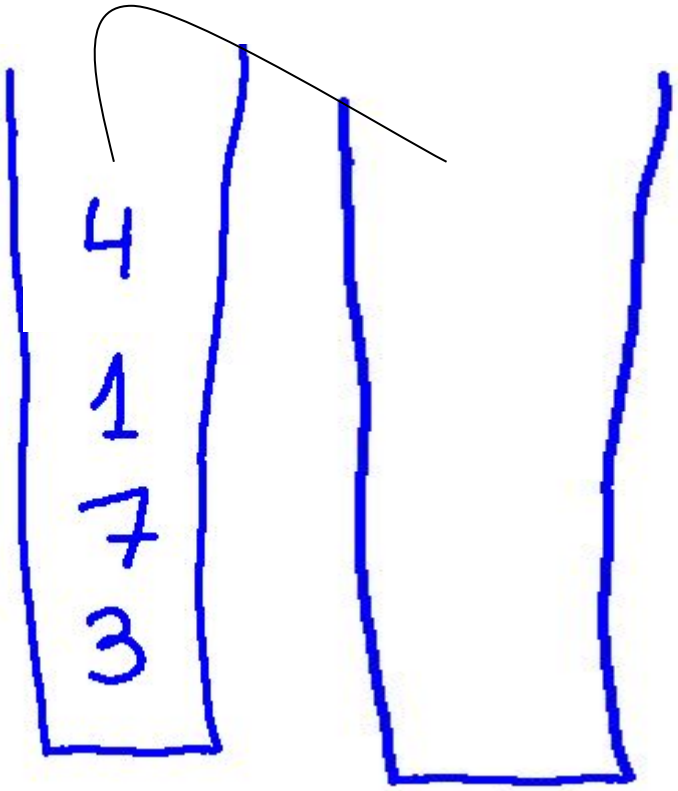
Origen

- ReadPila(nombrePila)
- InicPila(PilaX, ' ') ó InicPila(PilaX, ' 1 4 ')
- PilaVacia(nombrePila)
- Tope(nombrePila)
- Apilar(PilaX, Desapilar(PilaY))
- Apilar(PilaX, 8))
- WritePila(nombrePila)

•InicPila(Origen, ‘3 7 1 4 ')

```
Program Simple;
{Este Programa inicializa  una Pila y la muestra por pantalla}

{$INCLUDE /IntroProg/Estructu}

var
  Origen: pila;

Begin
   InicPila(Origen, '3 7 1 4 ');
   WritePila(Origen);
end.
```

Problema: pasar el tope de la pila Origen a la pila Destino



Origen Destino

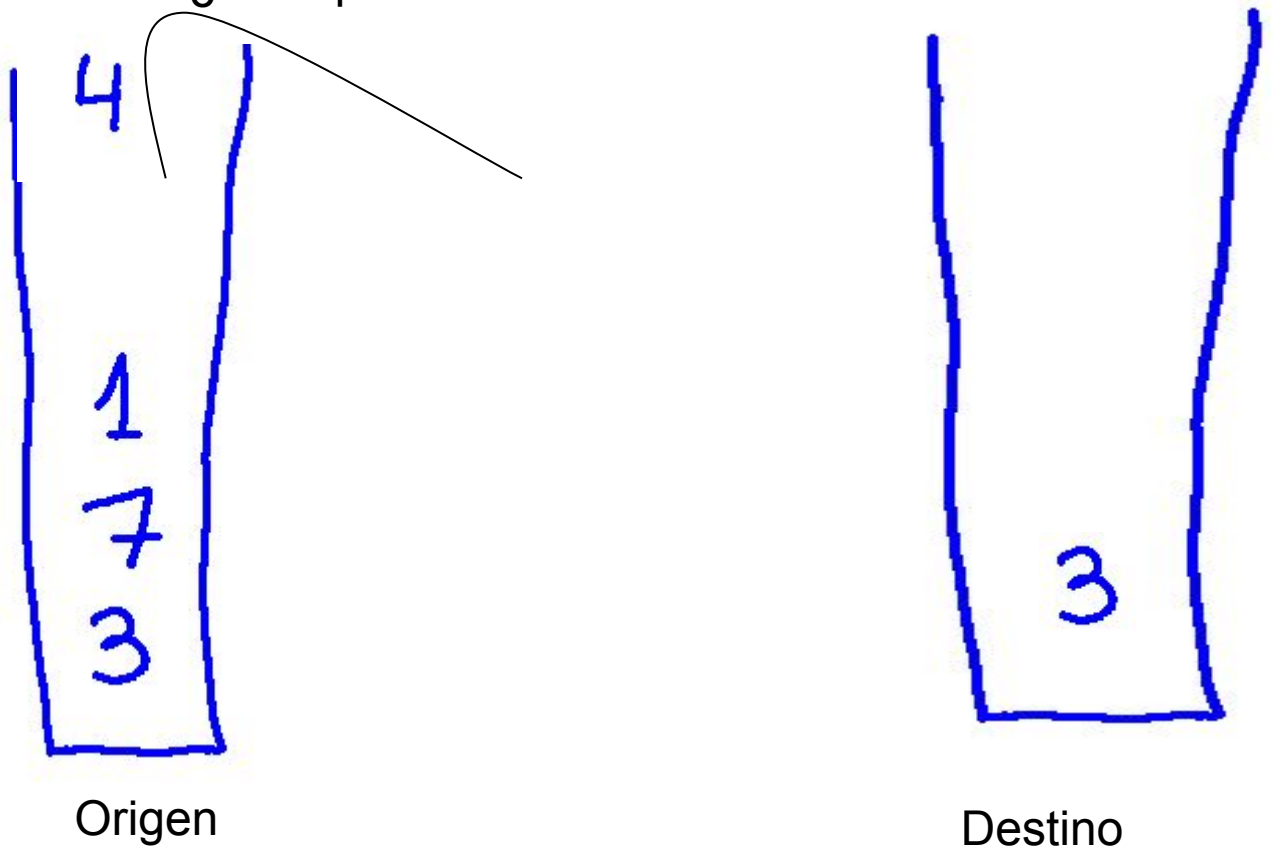
- ReadPila(nombrePila)
- InicPila(PilaX, ' ') ó InicPila(PilaX, '3 6 7 ')
- PilaVacia(nombrePila)
- Tope(nombrePila)
- Apilar(PilaX, Desapilar(PilaY))
- WritePila(nombrePila)

•Apilar(Destino, Desapilar(Origen);

```
Program PasaTope;
{Este Programa Leer los datos de una Pila y pasa el tope a otra Pila }
{$INCLUDE /IntroProg/Estructu}

var
  Origen, Destino: pila;

Begin
  ReadPila(Origen);      {se pide al usuario que entre datos para cargar  Pila Origen }
  InicPila(Destino, ' '); {se inicializa en Vacia la Pila Destino }
  Apilar (Destino, Desapilar(Origen));        {se pasa el tope de Origen a    Destino}
  WritePila(Origen);
   WritePila(Destino);
end.
```

Problema: pasar el tope de la pila Origen a la pila Destino

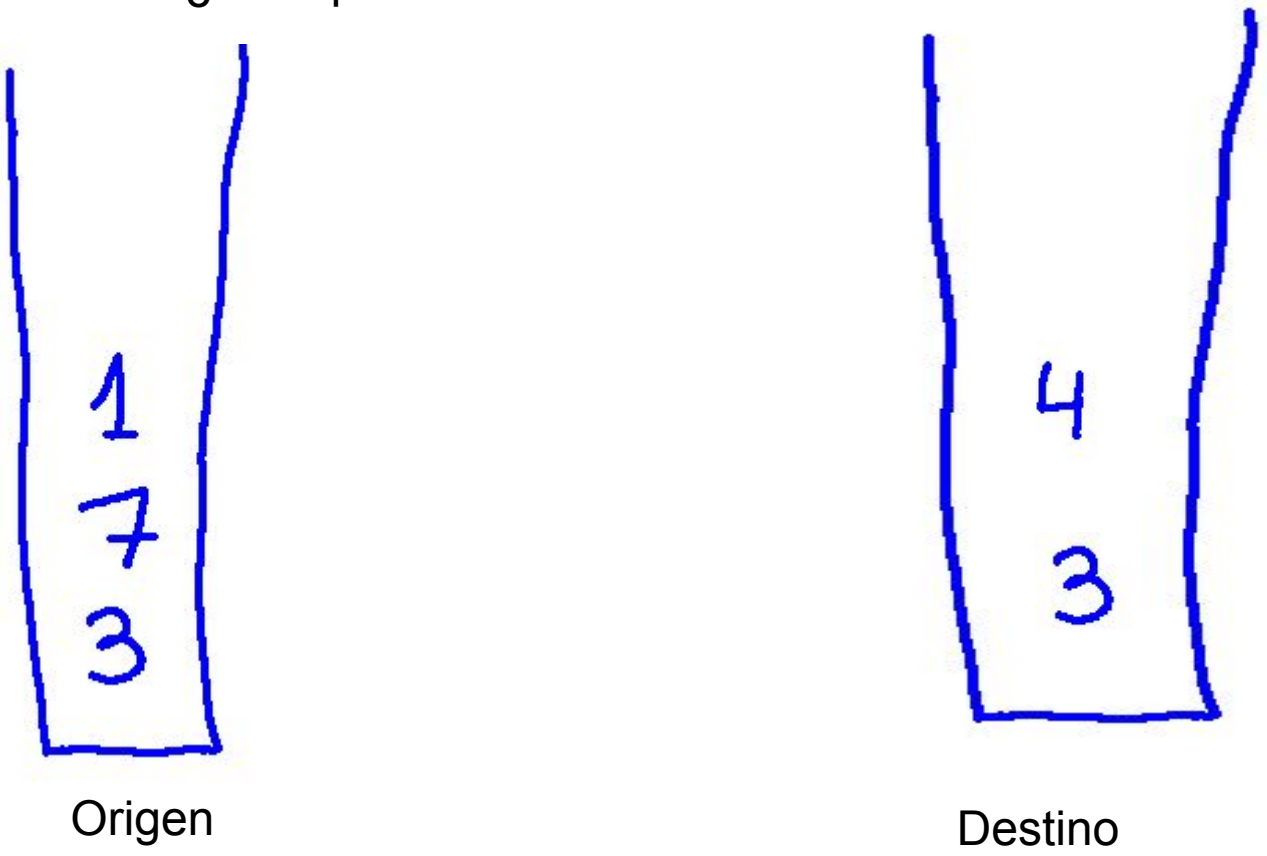
¿ Qué pasaría si Destino tiene elementos?



Apilar (Destino, Desapilar(Origen));

Problema: pasar el tope de la pila Origen a la pila Destino

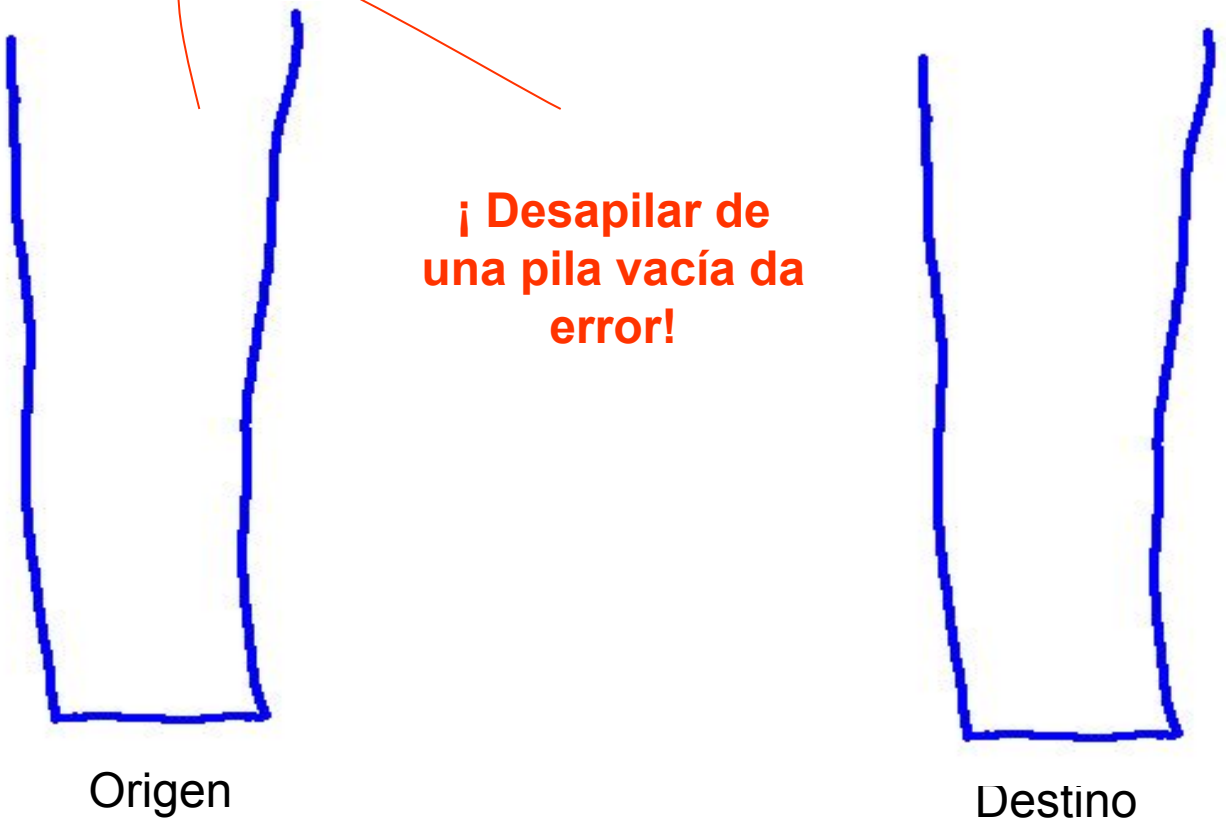
¿ Qué pasaría si Destino tiene elementos?



Apilar (Destino, Desapilar(Origen));

Problema: pasar el tope de la pila Origen a la pila Destino

¿Qué pasaría si Origen NO tiene elementos?



Apilar (Destino, Desapilar(Origen));

```
Program PasaTope;
{Este Programa Leer los datos de una Pila y pasa el tope a otra Pila }
{$INCLUDE /IntroProg/Estructu}

var
  Origen, Destino: pila;

Begin
  ReadPila(Origen);      {se pide al usuario que entre datos para cargar  Pila Origen }
  InicPila(Destino, ' '); {se inicializa en Vacia la Pila Destino }
  Apilar (Destino, Desapilar(Origen));        {se pasa el tope de Origen a     Destino}
  WritePila(Origen);
   WritePila(Destino);
end.
```

Si Pila Origen está vacía el programa termina la ejecución dando un error

¿Cómo se puede controlar?

Sentencia de control: Selección

A través de la selección se incorpora la capacidad de decisión en un programa.

IF (condición) **then**
acción o acciones a realizar si la condición es verdadera
else
acción o acciones a realizar si la condición es falsa

donde condición es una expresión que al ser evaluada puede tomar solamente uno de dos valores posibles: **verdadero** o **falso**.

De esta forma será posible seleccionar una de dos alternativas de acción posibles durante la ejecución del programa.

Sentencia de control: Selección

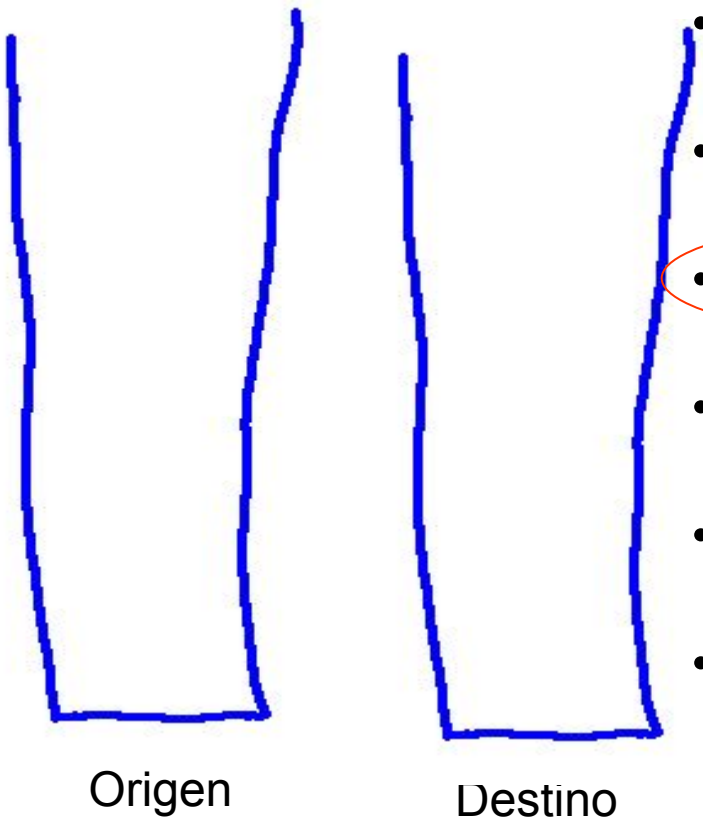
If (condición) **then**

acción o acciones a realizar si la condición es verdadera

En este caso, si la condición es falsa, no se especifica ningún camino alternativo a seguir.

Problema: pasar el tope de la pila Origen a la pila Destino

¿Qué pasaría si Origen NO tiene elementos?



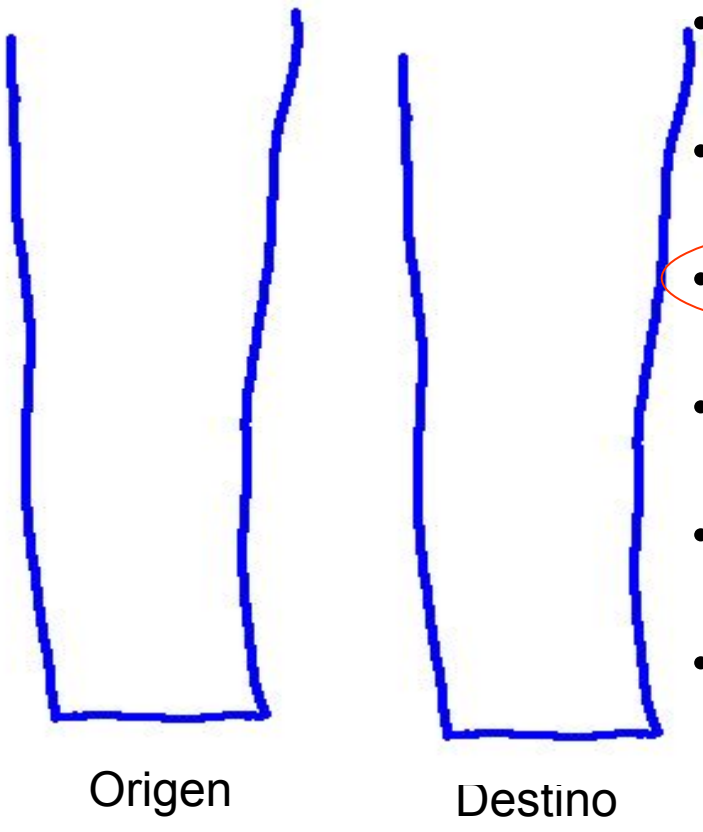
- ReadPila(nombrePila)
- InicPila(PilaX, ' ') ó InicPila(PilaX, '3 6 7 ')
- PilaVacia(nombrePila)
- Tope(nombrePila)
- Apilar(PilaX, Desapilar(PilaY))
- WritePila(nombrePila)

IF (condición) **then**

Apilar (Destino, Desapilar(Origen));

Problema: pasar el tope de la pila Origen a la pila Destino

¿Qué pasaría si Origen NO tiene elementos?

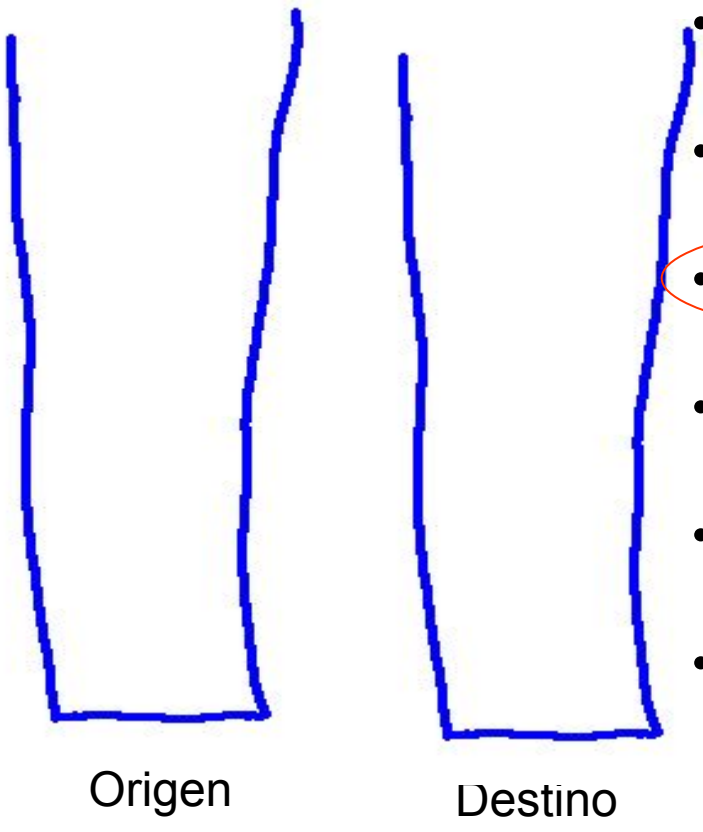


- ReadPila(nombrePila)
- InicPila(PilaX, ' ') ó InicPila(PilaX, '3 6 7 ')
- PilaVacia(nombrePila)
- Tope(nombrePila)
- Apilar(PilaX, Desapilar(PilaY))
- WritePila(nombrePila)

```
IF (PilaVacia(Origen)) then
        Apilar (Destino, Desapilar(Origen));
```

Problema: pasar el tope de la pila Origen a la pila Destino

¿Qué pasaría si Origen NO tiene elementos?



- ReadPila(nombrePila)
- InicPila(PilaX, ' ') ó InicPila(PilaX, '3 6 7 ')
- PilaVacia(nombrePila)
- Tope(nombrePila)
- Apilar(PilaX, Desapilar(PilaY))
- WritePila(nombrePila)

```
IF NOT (PilaVacia(Origen)) then
        Apilar (Destino, Desapilar(Origen));
```

C:\fpc\bin\PasaTope

```
Program PasaTope;
 {Este Programa pasa el tope de la pila Origen a Destino }

{$INCLUDE /IntroProg/Estructu}

var
  Origen, Destino: pila;

Begin
  ReadPila(Origen);
  InicPila(Destino, ' ');
  if not PilaVacia(Origen) then
      Apilar (Destino, Desapilar(Origen));
  WritePila(Origen);
  WritePila(Destino);
end.
```

Condición que dará Verdadero o Falso

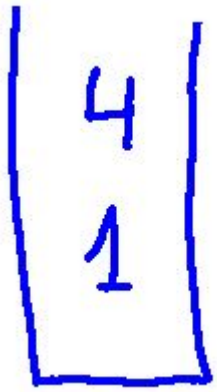
Estado de las Pilas durante ejecución:

Origen

Destino

```
Program PasaTope;
 {Este Programa pasa el tope de la pila Origen a Destino }

{$INCLUDE /IntroProg/Estructu}

var
  Origen, Destino: pila;

Begin
  ReadPila(Origen);
  InicPila(Destino, ' ');
  if not PilaVacia(Origen) then
     Apilar (Destino, Desapilar(Origen));
  WritePila(Origen);
  WritePila(Destino);
end.
```

```
C:\fpc\bin\PasaTope

Ingresar elementos a la Pila:
<Base><...><Tope>
      1      4
```



Estado de las Pilas durante ejecución:

Origen 1 4

Destino

```
Program PasaTope;
 {Este Programa pasa el tope de la pila Origen a Destino }

{$INCLUDE /IntroProg/Estructu}
var
  Origen, Destino: pila;

Begin
  ReadPila(Origen);
  InicPila(Destino, ' ');
  if not PilaVacia(Origen) then
      Apilar (Destino, Desapilar(Origen));
  WritePila(Origen);
  WritePila(Destino);
end.
```

```
C:\fpc\bin\PasaTope

Ingresar elementos a la Pila:
<Base><...><Tope>
      1      4
```

Estado de las Pilas durante ejecución:

Origen 1 4

Destino ‘
‘

```
Program PasaTope;
 {Este Programa pasa el tope de la pila Origen a Destino }

{$INCLUDE /IntroProg/Estructu}
var
  Origen, Destino: pila;

Begin
  ReadPila(Origen);
  InicPila(Destino, ' ');
  if not PilaVacia(Origen) then
      Apilar (Destino, Desapilar(Origen));
  WritePila(Origen);
  WritePila(Destino);
end.
```

C:\fpc\bin\PasaTope

Ingresar elementos a la Pila:
<Base><...><Tope>

1 4

Estado de las Pilas durante ejecución:

Origen 1 4

Destino
‘ ‘

```
Program PasaTope;
 {Este Programa pasa el tope de la pila Origen a Destino }

{$INCLUDE /IntroProg/Estructu}
var
  Origen, Destino: pila;

Begin
  ReadPila(Origen);
  InicPila(Destino, ' ');
  if not PilaVacia(Origen) then
     Apilar (Destino, Desapilar(Origen));
  WritePila(Origen);
  WritePila(Destino);
end.
```

C:\fpc\bin\PasaTope

Ingresar elementos a la Pila:
<Base><...><Tope>

1 4

Estado de las Pilas durante ejecución:

Origen 1

Destino 4

```
Program PasaTope;
 {Este Programa pasa el tope de la pila Origen a Destino }

{$INCLUDE /IntroProg/Estructu}
var
  Origen, Destino: pila;

Begin
  ReadPila(Origen);
  InicPila(Destino, ' ');
  if not PilaVacia(Origen) then
      Apilar (Destino, Desapilar(Origen));
  WritePila(Origen);
  WritePila(Destino);
end.
```

```
C:\fpc\bin\PasaTope

Ingresar elementos a la Pila:
<Base><...><Tope>

      1      4
<Base>    1       <Tope>
```

Estado de las Pilas durante ejecución:

Origen 1

Destino 4

```
Program PasaTope;
 {Este Programa pasa el tope de la pila Origen a Destino}

{$INCLUDE /IntroProg/Estructu}
var
  Origen, Destino: pila;

Begin
  ReadPila(Origen);
  InicPila(Destino, ' ');
  if not PilaVacia(Origen) then
      Apilar (Destino, Desapilar(Origen));
  WritePila(Origen);
  WritePila(Destino);
end.
```

C:\fpc\bin\PasaTope

Ingresar elementos a la Pila:
<Base><...><Tope>

1 4
<Base> 1 <Tope>
<Base> 4 <Tope>

Estado de las Pilas durante ejecución:

Origen 1

Destino 4

```
Program PasaTope;
 {Este Programa pasa el tope de la pila Origen a Destino
y avisa si la Pila Origen es vacia}

{$INCLUDE /IntroProg/Estructu}
var
  Origen, Destino: pila;

Begin
  ReadPila(Origen);
  InicPila(Destino, ' ');
  if not PilaVacia(Origen) then
        begin
            Apilar (Destino, Desapilar(Origen));
            WritePila(Origen);
            WritePila(Destino)
        end
     else
            Writeln('La Pila Origen está vacía');

end.
```

C:\fpc\bin\PasaTope

Ingresar elementos a la Pila:
<Base><...><Tope>

Estado de las Pilas durante ejecución:

Origen

Destino

```
Program PasaTope;
 {Este Programa pasa el tope de la pila Origen a Destino
y avisa si la Pila Origen es vacia}

{$INCLUDE /IntroProg/Estructu}
var
  Origen, Destino: pila;

Begin
  ReadPila(Origen);
  InicPila(Destino, ' ');
  if not PilaVacia(Origen) then
        begin
            Apilar (Destino, Desapilar(Origen));
            WritePila(Origen);
            WritePila(Destino)
        end
     else
            Writeln('La Pila Origen está vacía');

end.
```

C:\fpc\bin\PasaTope

Ingresar elementos a la Pila:
<Base><...><Tope>

Estado de las Pilas durante ejecución:

Origen

Destino

```
Program PasaTope;
 {Este Programa pasa el tope de la pila Origen a Destino
y avisa si la Pila Origen es vacia}

{$INCLUDE /IntroProg/Estructu}
var
  Origen, Destino: pila;

Begin
  ReadPila(Origen);
  InicPila(Destino, ' ');
  if not PilaVacia(Origen) then
        begin
            Apilar (Destino, Desapilar(Origen));
            WritePila(Origen);
            WritePila(Destino)
        end
     else
            Writeln('La Pila Origen está vacía');

end.
```

```
C:\fpc\bin\PasaTope

Ingresar elementos a la Pila:
<Base><...><Tope>
```

Estado de las Pilas durante ejecución:

Origen

Destino

```
Program PasaTope;
 {Este Programa pasa el tope de la pila Origen a Destino
y avisa si la Pila Origen es vacia}

uses Estructu;
var
  Origen, Destino: pila;

Begin
  ReadPila(Origen);
  InicPila(Destino, ' ');
  if not PilaVacia(Origen) then
        begin
            Apilar (Destino, Desapilar(Origen));
            WritePila(Origen);
            WritePila(Destino)
        end
     else
            Writeln('La Pila Origen está vacía');

end.
```

```
C:\fpc\bin\PasaTope

Ingresar elementos a la Pila:
<Base><...><Tope>



La Pila Origen está vacía
```

Estado de las Pilas durante ejecución:

Origen

Destino

```
Program PasaTope;
 {Este Programa pasa el tope de la pila Origen a Destino
y avisa si la Pila Origen es vacia}

uses Estructu;
var
  Origen, Destino: pila;

Begin
  ReadPila(Origen);
  InicPila(Destino, ' ');
  if not PilaVacia(Origen) then
      begin
          Apilar (Destino, Desapilar(Origen));
          WritePila(Origen);
          WritePila(Destino)
      end
    else
        begin
            Writeln('La Pila Origen está vacía');
        end;
  end.
```

Práctica

Analice la siguiente porción de código:

```
……….
        IF  tope(Pila1) < 10
                then   apilar(Resultado, desapilar(Pila1))
                else    apilar(OtroResultado, desapilar(Pila1));
……..
```

Suponiendo que las pilas Resultado y OtroResultado se inician vacías;
Mostrar como quedarian luego de ejecutar esta porción de código para:

a) Pila1 <base> 6 10 19 <tope>

Práctica

Analice la siguiente porción de código:

```
……….
        IF  tope(Pila1) < 10
                then   apilar(Resultado, desapilar(Pila1))
                else    apilar(OtroResultado, desapilar(Pila1));
……..
```

Suponiendo que las pilas Resultado y OtroResultado se inician vacías;
Mostrar como quedarian luego de ejecutar esta porción de código para:

a) Pila1 <base> 6 10 19 <tope>

Pila1 <base> 6 10 <tope>
OtroResultado <base> 19 <tope>
Resultado <base> <Vacia> <tope>

Práctica

Analice la siguiente porción de código:

```
……….
        IF  tope(Pila1) < 10
                then   apilar(Resultado, desapilar(Pila1))
                else    apilar(OtroResultado, desapilar(Pila1));
……..
```

Suponiendo que las pilas Resultado y OtroResultado se inician vacías;
Mostrar como quedarian luego de ejecutar esta porción de código para:

a) Pila1 <base> 6 10 8 <tope>

Práctica

Analice la siguiente porción de código:

```
……….
        IF  tope(Pila1) < 10
                then   apilar(Resultado, desapilar(Pila1))
                else    apilar(OtroResultado, desapilar(Pila1));
……..
```

Suponiendo que las pilas Resultado y OtroResultado se inician vacías;
Mostrar como quedarian luego de ejecutar esta porción de código para:

a) Pila1 <base> 6 10 8 <tope>

Pila1 <base> 6 10 <tope>
OtroResultado <base> <Vacia> <tope>
Resultado <base> 8 <tope>

Problema:

Pasar todos los elementos de una Pila a Otra

```
Program PasaPila;
{Este Programa pasa los elementos de la pila Origen a Destino}



{$INCLUDE /IntroProg/Estructu}
var
  Origen, Destino: pila;

Begin
  ReadPila(Origen);
  InicPila(Destino, ' ');
  if not PilaVacia(Origen) then
    Apilar (Destino, Desapilar(Origen));
  if not PilaVacia(Origen) then
    Apilar (Destino, Desapilar(Origen));
 if not PilaVacia(Origen) then
    Apilar (Destino, Desapilar(Origen));
……
  WritePila(Origen);
  WritePila(Destino)

end.
```

Estructura de control: Iteración (WHILE)

- **Existen situaciones en las que se quiere repetir un conjunto de acciones pero se desconoce de antemano el número de veces.**

  La iteración es una estructura de control que permite al programa ejecutar en forma repetitiva un conjunto de acciones utilizando una condición para indicar su finalización.

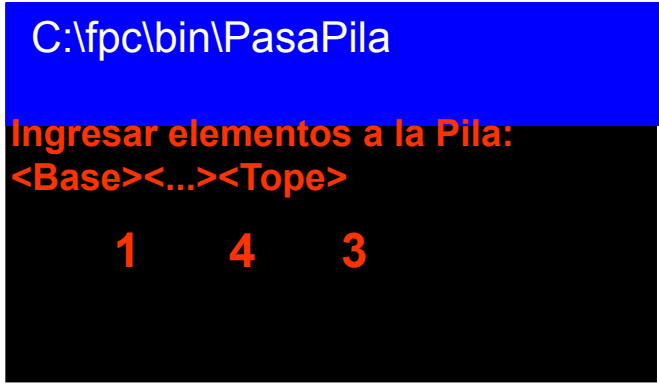
While (condición) do

Instrucción o instrucciones

```
Program PasaPila;
{Este Programa pasa los elementos de la pila Origen a Destino}



{$INCLUDE /IntroProg/Estructu}
var
  Origen, Destino: pila;

Begin
  ReadPila(Origen);
  InicPila(Destino, ' ');
  if not PilaV a(Orige hen
    Apilar (Des , D pilar(Origen));
  if not PilaVacia en) then
    Apilar (Destin sapilar(Origen));
 if not PilaVaci ri ) then
    Apilar (De o, De ilar(Origen));
……
  WritePila(Origen);
  WritePila(Destino)

end.
```

While not PilaVacia(Origen) do
** Apilar (Destino, Desapilar(Origen));**

```
Program PasaPila;
{Este Programa pasa los elementos de la pila
Origen a Destino}


{$INCLUDE /IntroProg/Estructu}
var
  Origen, Destino: pila;

Begin
  ReadPila(Origen);
  InicPila(Destino, ' ');
  While not PilaVacia(Origen) do
       Apilar (Destino, Desapilar(Origen));
  WritePila(Origen);
  WritePila(Destino)
end.
```



Estado de las Pilas durante ejecución:

Origen 1 4 3

Destino

```
Program PasaPila;
{Este Programa pasa los elementos de la pila
Origen a Destino}


{$INCLUDE /IntroProg/Estructu}
var
  Origen, Destino: pila;

Begin
  ReadPila(Origen);
  InicPila(Destino, ' ');
  While not PilaVacia(Origen) do
      Apilar (Destino, Desapilar(Origen));
  WritePila(Origen);
  WritePila(Destino)
end.
```

```
C:\fpc\bin\PasaPila

Ingresar elementos a la Pila:
<Base><...><Tope>
      1     4     3
```

Estado de las Pilas durante ejecución:

Origen	1 4 3
Destino	‘ ‘

```
Program PasaPila;
{Este Programa pasa los elementos de la pila
Origen a Destino}


{$INCLUDE /IntroProg/Estructu}
var
  Origen, Destino: pila;

Begin
  ReadPila(Origen);
  InicPila(Destino, ' ');
  While not PilaVacia(Origen) do
      Apilar (Destino, Desapilar(Origen));
  WritePila(Origen);
  WritePila(Destino)
end.
```

```
C:\fpc\bin\PasaPila

Ingresar elementos a la Pila:
<Base><...><Tope>
      1     4     3
```

Estado de las Pilas durante ejecución:

Origen	1 4 3
Destino	‘ ‘

```
Program PasaPila;
{Este Programa pasa los elementos de la pila
Origen a Destino}


{$INCLUDE /IntroProg/Estructu}
var
  Origen, Destino: pila;

Begin
  ReadPila(Origen);
  InicPila(Destino, ' ');
  While not PilaVacia(Origen) do
       Apilar (Destino, Desapilar(Origen));
  WritePila(Origen);
  WritePila(Destino)
end.
```

```
C:\fpc\bin\PasaPila

Ingresar elementos a la Pila:
<Base><...><Tope>
      1     4     3
```

Estado de las Pilas durante ejecución:

Origen 1 4

Destino 3

```
Program PasaPila;
{Este Programa pasa los elementos de la pila
Origen a Destino}


{$INCLUDE /IntroProg/Estructu}
var
  Origen, Destino: pila;

Begin
  ReadPila(Origen);
  InicPila(Destino, ' ');
  While not PilaVacia(Origen) do
      Apilar (Destino, Desapilar(Origen));
  WritePila(Origen);
  WritePila(Destino)
end.
```

```
C:\fpc\bin\PasaPila

Ingresar elementos a la Pila:
<Base><...><Tope>
      1     4     3
```

Estado de las Pilas durante ejecución:

Origen 1 4

Destino 3

```
Program PasaPila;
{Este Programa pasa los elementos de la pila
Origen a Destino}


{$INCLUDE /IntroProg/Estructu}
var
  Origen, Destino: pila;

Begin
  ReadPila(Origen);
  InicPila(Destino, ' ');
  While not PilaVacia(Origen) do
      Apilar (Destino, Desapilar(Origen));
  WritePila(Origen);
  WritePila(Destino)
end.
```

```
C:\fpc\bin\PasaPila

Ingresar elementos a la Pila:
<Base><...><Tope>
      1     4     3
```

Estado de las Pilas durante ejecución:

Origen 1

Destino 3 4

```
Program PasaPila;
{Este Programa pasa los elementos de la pila
Origen a Destino}


{$INCLUDE /IntroProg/Estructu}
var
  Origen, Destino: pila;

Begin
  ReadPila(Origen);
  InicPila(Destino, ' ');
  While not PilaVacia(Origen) do
      Apilar (Destino, Desapilar(Origen));
  WritePila(Origen);
  WritePila(Destino)
end.
```

```
C:\fpc\bin\PasaPila

Ingresar elementos a la Pila:
<Base><...><Tope>
      1     4     3
```

Estado de las Pilas durante ejecución:

Origen 1

Destino 3 4

```
Program PasaPila;
{Este Programa pasa los elementos de la pila
Origen a Destino}


{$INCLUDE /IntroProg/Estructu}
var
  Origen, Destino: pila;

Begin
  ReadPila(Origen);
  InicPila(Destino, ' ');
  While not PilaVacia(Origen) do
      Apilar (Destino, Desapilar(Origen));
  WritePila(Origen);
  WritePila(Destino)
end.
```

```
C:\fpc\bin\PasaPila

Ingresar elementos a la Pila:
<Base><...><Tope>
      1     4     3
```

Estado de las Pilas durante ejecución:

Origen

Destino 3 4 1

```
Program PasaPila;
{Este Programa pasa los elementos de la pila
Origen a Destino}


{$INCLUDE /IntroProg/Estructu}
var
  Origen, Destino: pila;

Begin
  ReadPila(Origen);
  InicPila(Destino, ' ');
  While not PilaVacia(Origen) do
      Apilar (Destino, Desapilar(Origen));
  WritePila(Origen);
  WritePila(Destino)
end.
```

```
C:\fpc\bin\PasaPila

Ingresar elementos a la Pila:
<Base><...><Tope>
      1     4     3
```

Estado de las Pilas durante ejecución:

Origen

Destino 3 4 1

```
Program PasaPila;
{Este Programa pasa los elementos de la pila
Origen a Destino}


{$INCLUDE /IntroProg/Estructu}
var
  Origen, Destino: pila;

Begin
  ReadPila(Origen);
  InicPila(Destino, ' ');
  While not PilaVacia(Origen) do
       Apilar (Destino, Desapilar(Origen));
  WritePila(Origen);
  WritePila(Destino)
end.
```

```
C:\fpc\bin\PasaPila

Ingresar elementos a la Pila:
<Base><...><Tope>

      1     4     3
<Base> Vacía    <Tope>
```

Estado de las Pilas durante ejecución:

Origen

Destino 3 4 1

```
Program PasaPila;
{Este Programa pasa los elementos de la pila
Origen a Destino}


{$INCLUDE /IntroProg/Estructu}
var
  Origen, Destino: pila;

Begin
  ReadPila(Origen);
  InicPila(Destino, ' ');
  While not PilaVacia(Origen) do
       Apilar (Destino, Desapilar(Origen));
  WritePila(Origen);
  WritePila(Destino)
end.
```

```
C:\fpc\bin\PasaPila

Ingresar elementos a la Pila:
<Base><...><Tope>
      1     4     3
<Base>  Vacía    <Tope>
<Base>    3 4 1  <Tope>
```

Estado de las Pilas durante ejecución:

Origen

Destino 3 4 1

```
Program PasaPila;
{Este Programa pasa los elementos de la pila
Origen a Destino}


{$INCLUDE /IntroProg/Estructu}
var
  Origen, Destino: pila;

Begin
  ReadPila(Origen);
  InicPila(Destino, ' ');
  While not PilaVacia(Origen) do
      Apilar (Destino, Desapilar(Origen));
  WritePila(Origen);
  WritePila(Destino)
end.
```

Al ir desapilando los elementos de Origen, en algún momento la pila Origen estará vacía y la condición se dejará de cumplir

```
Program PasaPila;
{Este Programa ¿pasa los elementos de la pila Origen a Destino?}


{$INCLUDE /IntroProg/Estructu}
var
  Origen, Destino: pila;

Begin
  ReadPila(Origen);
  InicPila(Destino, ' ');
  While not PilaVacia(Origen) do
          WritePila (Origen);
  WritePila(Origen);
  WritePila(Destino)
end.
```

¡INCORRECTO!

Nunca va a ser falsa la condición, no se termina la iteración

Estructura de control: Iteración (WHILE)

While (condición sea verdadera) **do**
Acción o acciones a realizar

- La condición es evaluada en cada ciclo

- Debe haber alguna acción dentro del conjunto de acciones que modifiquen el valor de la condición (SE CORTA la Iteración)

Ejercitación

Pila1 y Pila2 son pilas.

a) ¿Cuál es la condición del siguiente ciclo?

b)¿Cuándo finaliza el ciclo?

```
while not pilaVacia(Pila1) do
 begin
    apilar (Pila2, desapilar(Pila1))
  end
```

Ejercitación 2

Pila1, Pila2, y Descarte son pilas. Pila1 y Plla 2 tienen elementos y Descarte está vacia

a) ¿Cuál es la condición del siguiente ciclo?
b)¿Cuándo finaliza el ciclo?

```
while not pilaVacia(Pila1) do
 begin
    apilar (Descarte, desapilar(pila2))
  end
```

MAL

Resumen: operaciones con Pilas

- ReadPila(nombrePila)
- InicPila(nombrePila, ' ') o InicPila(nombrePila, ‘3 6 7 ')
- PilaVacia(nombrePila)
- Tope(nombrePila)
- Apilar(nombrePila,Desapilar(nombreOtra Pila)) o
  Apilar(nombrePila, 8)
- WritePila(nombrePila)

Estructuras de control

- **SECUENCIA**

< sentencia 1 >;√

< sentencia 2 >;√

< sentencia 3 >;√

Se ejecutan una sola vez en el mismo orden en que aparecen

Estructuras de control

- **SELECCIÓN**

if <condición> then
< sentencia si condición es verdadera >;

if <condición> then
< sentencia si condición es verdadera >
else
< sentencia si condición es falsa >;

Estructuras de control

ITERACIÓN

while <condición> do

< sentencia mientras condición sea verdadera >;

En <sentencia> se debe asegurar que la condición va a cambiar para ser falsa en algún momento y cortar el ciclo

SENTENCIAS COMPUESTAS

Una serie de instrucciones básicas que se ejecutan una a continuación de otra como un bloque.

```
BEGIN
    <SENTENCIA 1>;
    <SENTENCIA 2>
END
```

Estructuras de control

```
if <condición> then
begin
      < sentencia si condición es verdadera >;
      < sentencia si condición es verdadera >;
      < sentencia si condición es verdadera >
end
else
begin
      < sentencia si condición es falsa >;
      < sentencia si condición es falsa >
end
```

Estructuras de control

while <condición> do
begin

< sentencia mientras condición es verdadera >;
< sentencia mientras condición es verdadera >;
< sentencia mientras condición es verdadera >

end

SENTENCIAS COMPUESTAS

¿Qué diferencia hay entre los dos códigos?

```
.....
IF not PilaVacia(Origen) THEN
    IF tope(Origen) < 5 THEN
    BEGIN
      Apilar (Menores, Desapilar(Origen));
       writePila(Menores);
    END;
.....
```

```
.....
IF not PilaVacia(Origen) THEN BEGIN
    IF tope(Origen) < 5 THEN
          Apilar (Menores, Desapilar(Origen))
      writePila(Menores);
END
    .....
```

Estructuras de control

Las estructuras de control se pueden combinar

EJEMPLO

```
……..
While not PilaVacia(Origen) do
      if tope(Origen) < 1 then
      begin
            Apilar (Menores, Desapilar(Origen));
            writePila(Menores)
      end
      else
       begin
             Apilar (Mayores, Desapilar(Origen));
              writePila(Mayores)
       end
  ……….
```

INDENTACIÓN

```
Program DivideMenores5;
{este programa pasa todos los números
de la pila  Origen a la pila MenorIgual si son
menores o iguales a 5  y sino los pasa a
 Mayores}

{ $INLCUDE  /usr/Estructu }
var
  Origen, Menores, Mayores: Pila;

Begin
    ReadPila(Origen);
    InicPila(MenorIgual, ' ');
    InicPila(Mayores, ' ');
    While not PilaVacia(Origen) do
        if tope(Origen) =< 5 then
            Apilar (MenorIgual, Desapilar(Origen))
        else
            Apilar (Mayores, Desapilar(Origen));
    writePila(MenorIgual);
    writePila(Mayores);
end.
```

```
Program DivideMenores5;
{este programa pasa todos los números
de la pila  Origen a la pila MenorIgual si son
menores o iguales a 5  y sino los pasa a
Mayores}

{ $INLCUDE  /usr/Estructu }
var
Origen, Menores, Mayores: Pila;

Begin
ReadPila(Origen);
InicPila(MenorIgual, ' ');
InicPila(Mayores, ' ');
While not PilaVacia(Origen) do
if tope(Origen) =< 5 then
Apilar (MenorIgual, Desapilar(Origen))
else
Apilar (Mayores, Desapilar(Origen));
writePila(MenorIgual);
writePila(Mayores);
end.
```

INDENTACIÓN

```
Program DivideMenores5;
{este programa pasa todos los números
de la pila  Origen a la pila MenorIgual si son
menores o iguales a 5  y sino los paso a
Mayores}
{ $INLCUDE  /usr/Estructu }
var
  Origen, Menores, Mayores: Pila;

Begin
    ReadPila(Origen);
    InicPila(MenorIgual, ' ');
    InicPila(Mayores, ' ');
    While not PilaVacia(Origen) do
       if tope(Origen) =< 5 then
            Apilar (MenorIgual, Desapilar(Origen))
       else
            Apilar (Mayores, Desapilar(Origen));
    writePila(MenorIgual);
    writePila(Mayores);
end.
```

Se puede “ver” donde termina el conjunto de instrucciones involucradas en una sentencia de control

IMPORTANTE

Para usar Pilas en el entorno:

{$INCLUDE /IntroProg/Estructu}

Se puede guardar y ejecutar todas las veces que quiera: recordar poner al guardar .pas al archivo

Recordar que el entorno sólo reconoce un archivo .pas (se debe renombrar el resto)