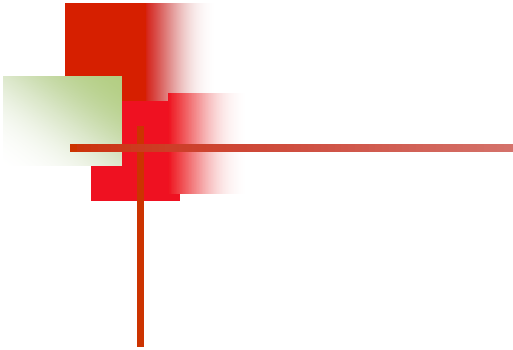


Tipo de dato Abstracto (Tad)

PILAS

COLAS

Tipos Abstractos de Datos - PILA

Poner

Tope

Sacar

Tipos Abstractos de Datos - PILA

El tipo de Dato PILA (stack) es una estructura de datos que organiza los datos de la siguiente manera:

A partir de una dirección de memoria, los datos se almacenan como si fueran una colección ordenada de elementos (cartas, platos, libros, camisas, etc).

El orden está asociado al orden de llegada a la estructura. Ejemplo

Tipos Abstractos de Datos - PILA

CARACTERÍSTICAS

- Homogénea: ya que almacena elementos del mismo tipo.

- Dinámica: ya que permite agregar y sacar elementos durante la ejecución del programa.

- Acceso LIFO: (Last In First Out): los elementos se recuperan en orden inverso al que fueron almacenados.

- En cualquier momento se puede recuperar el objeto que se encuentra al tope de la pila (es decir, el último que fue guardado).

- Son muy pocos los lenguajes de programación que cuentan con una estructura de este tipo como tipo estándar. En particular, Pascal no dispone del tipo de datos pila.

Tipos Abstractos de Datos - PILA

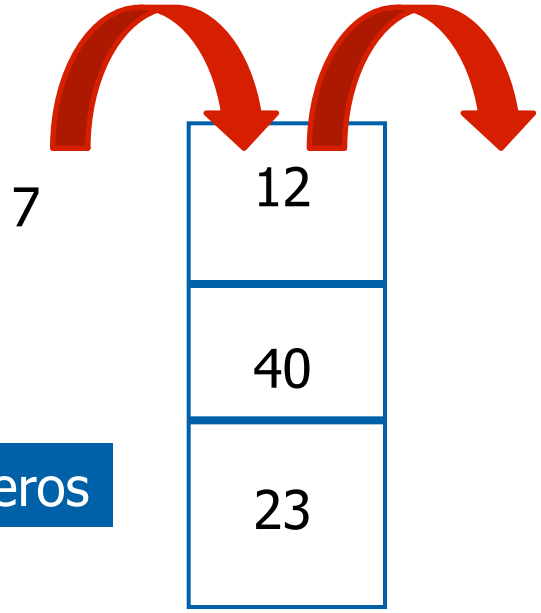
OPERACIONES

- Crear una pila.
- Agregar un elemento.
- Sacar un elemento.
- Poder saber cual es el elemento que está al tope de la pila.
- Cantidad de elementos de la pila.
- Saber si una pila está vacía.

Tipos Abstractos de Datos - PILA

Trabajaremos con el TAD PILA de Enteros

- ¿Por qué conviene definir el TAD Pila de enteros?
- ¿Cómo funciona el poner (Agregar) un elemento de la estructura?
- ¿Cómo funciona el Sacar un elemento de la estructura?



Pila de Enteros

¿Dónde se Agrega un dato?

¿De dónde se saca un dato?

Tipos Abstractos de Datos - PILA

Recordemos las operaciones:

- Crear una pila: st_Create
- Agregar un elemento de la pila: st_Push
- Sacar un elemento de la pila: st_Pop
- Conocer el elemento que está al tope de la pila: st_Top
- Cantidad de elementos de la pila: st_Length
- Saber si una pila está vacía: st_Empty

Tipos Abstractos de Datos - PILA

TAD TipoPILA;
Interface
  type exportado Pila;

Procedure st_create (var p:pila); //crea una pila vacía.

Procedure st_push (var p:pila; elem:integer);
//agrega el elemento elem a la pila p.

Procedure st_pop (var p:pila; var elem:integer);
//saca el elemento elem a la pila p. La pila debe tener elementos.

Procedure st_top (p:pila; var elem:integer);
//devuelve el elemento ubicado al tope de la pila sin sacarlo. La pila debe tener elementos.

Function st_length (p:pila):integer;//devuelve la cantidad de elementos de la pila p.

Fucntion st_empty (p:pila):boolean; //devuelve true si la pila p está vacía, false en caso contrario.

Tipos Abstractos de Datos - PILA

Implementacion

```
piladatos= ^nodo;

nodo =record
        elemento:integer;
        sig: piladatos;
      end;

pila= record;
        datos:piladatos;
        cant:integer;
      end;
```

depende de los elementos de la pila, en este caso es de enteros

Mantiene la cantidad de elementos de la pila, es para realizar más fácil la implementación de algunos módulos

Tipos Abstractos de Datos - PILA

```
Procedure st_create (var p:pila); //crea una pila vacía.
   Begin
      p.cant:= 0;
      p.datos:= nil;
   End;
```

Tipos Abstractos de Datos - PILA

```
Procedure st_push (var p: pila; elem: integer);
//agrega el elemento elem a la pila p.
Var
  aux:piladatos;
Begin
   new (aux);
   aux^.elemento:= elem;
   if (p.datos= nil) then
                          aux^.sig:=nil;
                        else begin
                          aux^.sig:= p.datos;
    p.datos:= aux;
    p.cant:= p.cant+1;
End;
```

Tipos Abstractos de Datos - PILA

```
Procedure st_pop (var p:pila; var elem:integer);
//saca y devuelve el elemento del tope de la pila p.
Var
  aux:piladatos;
Begin
  elem:= p.datos^.elemento;
  aux:= p.datos;
  p.datos:= p.datos^.sig;
  dispose (aux);
  p.cant:= p.cant-1;
End;
```

Notar que en este procedimiento **no se verifica** que la pila tenga elementos

Tipos Abstractos de Datos - PILA

```
Procedure st_top (p:pila; var elem:integer);
//devuelve el elemento del tope de la pila p, sin eliminar el elemento de la pila.

Begin
  elem:= p.datos^.elemento;
End;
```

Notar que en este procedimiento **no se verifica** que la pila tenga elementos

Tipos Abstractos de Datos - PILA

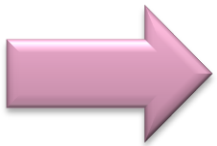
```
Function st_length (p: pila): integer;
//devuelve la cantidad de elementos de la pila.
Begin
  st_length:= p.cant;
End;
```

```
Function st_empty (p: pila): boolean;
//devuelve true si la pila está vacia, false en cc.
Begin
  st_empty:= (p.cant=0);
End;
```

Tipos Abstractos de Datos - PILA

Realice un programa que lea números enteros hasta que viene el número 999. Una vez que la lectura ha finalizado se debe imprimir los números leídos en orden inverso.

Si se lee 25 6 89 23 999



Se debería imprimir 23 89 6 25

Tipos Abstractos de Datos - EJEMPLO

```
Program uno;
Uses TipoPila;
Var
  p:pila;
  ele:integer;
Begin
   read(ele);
   st_create(p);
   while (ele <> 999) do begin
         st_push(p,ele);
         read (ele);
   end;
     while (not st_empty (p) ) do begin
           st_pop(p,ele);
           write(elem);
      end;
   end.
```

Indica que se utiliza el TAD

Variable del tipo exportado

Tipos Abstractos de Datos - COLA

El tipo de Dato COLA (queue) es una estructura de datos que organiza los datos de la siguiente manera:

- A partir de una dirección de memoria, los datos se almacenan sucesivamente como si fueran una colección ordenada de elementos (cartas, clientes frente a una ventanilla, libros en un estante, mensajes a una casilla de correo, etc), y

- En cualquier momento se puede recuperar el objeto que se encuentra primero en la estructura (es decir, el primero que fue guardado).

Tipos Abstractos de Datos - COLA

CARÁCTERÍSTICAS

- Homogénea: ya que almacena elementos del mismo tipo.

- Dinámica: ya que permite agregar y sacar elementos durante la ejecución del programa.

- Acceso FIFO: (First In First Out): los elementos se recuperan en orden inverso al que fueron almacenados.

- En cualquier momento se puede recuperar el elemento que se encuentra en el frente de la cola (es decir, el primero que fue guardado).

- Son muy pocos los lenguajes de programación que cuentan con una estructura de este tipo como tipo estándar. En particular, Pascal no dispone del tipo de datos cola.

Tipos Abstractos de Datos - COLA

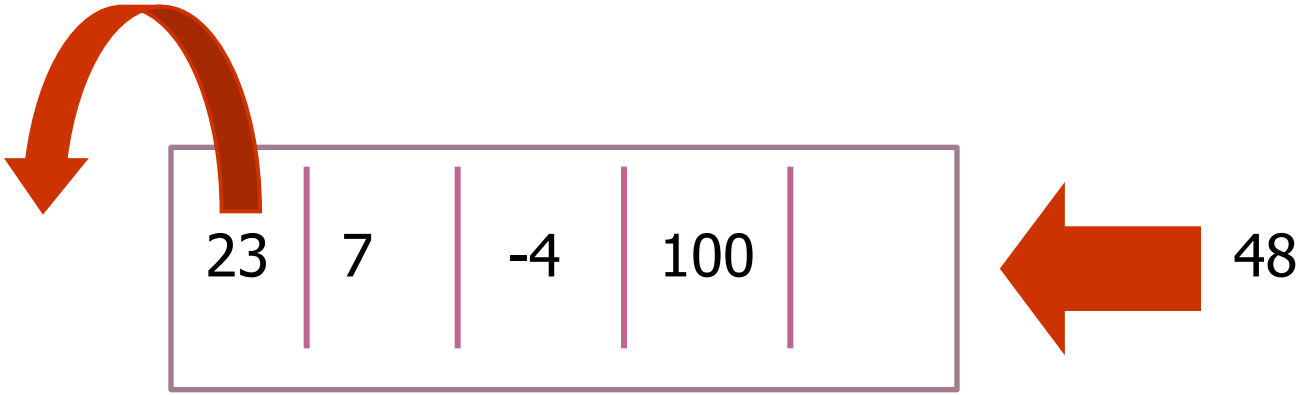
OPERACIONES

- Crear una cola.
- Agregar un elemento a la estructura.
- Sacar un elemento de la estructura.
- Poder saber cual es el elemento que está en el frente de la cola.
- Poder saber cual es el elemento que está en el fondo de la cola.
- Cantidad de elementos de la cola.
- Saber si una cola está vacía.

Tipos Abstractos de Datos - COLA

Trabajaremos con el TAD COLA de Enteros

- ¿Por qué conviene definir el TAD Cola de enteros?
- ¿Cómo funciona el poner (Agregar) un elemento de la estructura?
- ¿Cómo funciona el Sacar un elemento de la estructura?



Cola de Enteros

Tipos Abstractos de Datos - COLA

OPERACIONES

- Crear una cola: q_create
- Agregar un elemento a la estructura: q_push
- Sacar un elemento de la estructura: q_pop
- Poder saber cual es el elemento que está en el frente de la cola q_top
- Poder saber cual es el elemento que está en el fondo de la cola: q_bottom.
- Cantidad de elementos de la cola: q_length
- Saber si una cola está vacía: q_empty

Tipos Abstractos de Datos - COLA

```
TAD TipoCOLA;  //maneja una cola de enteros
Interface
  type exportado cola;

Procedure q_create (var c:cola); //crea una cola vacía.

Procedure q_push (var c:cola; elem:integer); //agrega el elemento elem a la cola c.

Procedure q_pop (var c:cola; var elem:integer); //saca el primer elemento de la cola p.

Procedure q_top (c:cola; var elem:integer); //devuelve el elemento ubicado en el frente de la cola sin sacarlo.

Procedure q_botton (c:cola; var elem:integer); //devuelve el elemento ubicado en el fondo de la cola sin sacarlo.

Function q_length (c:cola):integer; //devuelve la cantidad de elementos de la cola c.

Function q_empty (c:cola):boolean; //devuelve true si la cola p está vacía, false en caso contrario.
```

Tipos Abstractos de Datos - COLA

Discutamos una posible implementación...

```
Implementacion

  coladatos= ^nodo;

  nodo =record
            elemento:integer;
            sig: coladatos;
          end;

  cola= record;
          pri:coladatos;
          ult:coladatos;
          cant:integer;
        end;
```

- ¿Podemos utilizar la misma representación que en el Tad pila?
- ¿Cómo conviene representarla?
- ¿Por qué?

depende de los elementos de la cola

Mantiene la cantidad de elementos de la cola.

Tipos Abstractos de Datos - COLA

```
Procedure q_create (var c:cola); //crea una cola vacía.
Begin
  c.cant:= 0;
  c.pri:= nil;
  c.ult:= nil;
End;
```

Tipos Abstractos de Datos - COLA

```
Procedure q_push (var c:cola; elem:integer);
//agrega el elemento elem a la cola c.
Var
  aux:coladatos;

Begin
   new (aux);
   aux^.elemento:= elem;
   aux^.sig:= nil;
   if (c.pri= nil) then  begin
                          c.pri:= aux;
                          c.ult:=aux;
                        end
                        else begin
                          c.ult^.sig:= aux;
                          c.ult:= aux;
                         end;
  c.cant:= c.cant+1;
End;
```

Tipos Abstractos de Datos - COLA

```
Procedure q_pop (var c:cola; var elem:integer);
//saca y devuelve el elemento del tope de la cola c.
Var
  aux:coladatos;
Begin
  elem:= c.pri^.elemento;
  aux:= c.pri;
  c.pri:= c.pri^.sig;
  dispose (aux);
  c.cant:= c.cant-1;
End;
```

Notar que en este procedimiento **no se verifica** que la cola tenga elementos

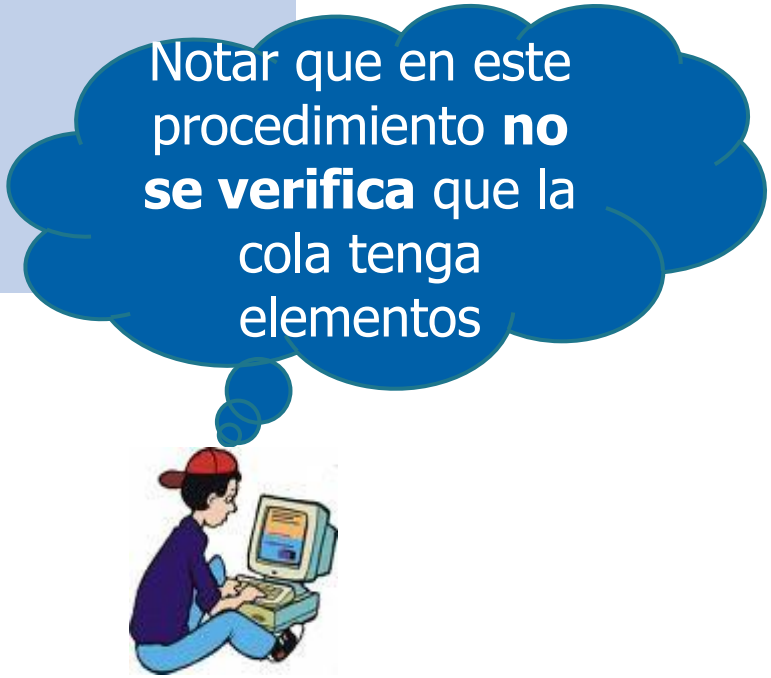
Tipos Abstractos de Datos - COLA

```
Procedure q_top (c:cola; var elem:integer);
//devuelve el elemento del tope de la cola c, sin eliminar el elemento de la cola.

Begin
  elem:= c.pri^.elemento;
End;
```

Notar que en este procedimiento **no se verifica** que la cola tenga elementos

Tipos Abstractos de Datos - COLA

```
Procedure q_botton (c:cola; var elem:integer);
//devuelve el elemento del final de la cola c, sin eliminar el elemento de la cola.

Begin
  elem:= c.ult^.elemento;
End;
```



Tipos Abstractos de Datos - COLA

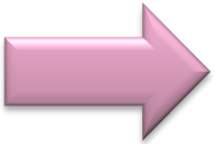
```
Function q_length (c: cola): integer;
 //devuelve la cantidad de elementos de la cola.

Begin
  q_length:= c.cant;
End;
```

```
Function q_empty (c: cola): boolean;
//devuelve true si la cola está vacia, false en cc.

Begin
  q_empty:= (c.cant=0);
End;
```

Tipos Abstractos de Datos – Ejemplo

Realice un programa que lea dos secuencias de números y las almacene. Cada secuencia termina con el número 0. Al finalizar toda la lectura informar si las secuencias son simétricas.

Si se lee 25 6 89 23 29 0 29 23 89 6 25



Se debería imprimir las secuencias son simétricas

Analicemos cual es la ventaja de utilizar las estructuras vistas...

Tipos Abstractos de Datos - Ejemplo

```
Program uno;
Uses TipoPila, TipoCola;
Var
  p:pila;  c:cola;

Begin
  st_create(p);
  q_create(c);
  cargarPila (p);
  cargarCola (c);
  if (simetricas (p, c)) then write (“Son Simétricas”)
                         else write (“No son Simétricas”);
End.
```

Tipos Abstractos de Datos - Ejemplo

```
Procedure cargarPila (var p:pila);
 var
   elem:integer;
 begin
   read (elem);
   while (elem <> 0) do begin
         st_push(p, elem);
         read (elem);
   end;
End;
```

Tipos Abstractos de Datos - Ejemplo

```
Procedure cargarCola (var c:cola);
 var
   elem:integer;
 begin
   read (elem);
   while (elem <> 0) do begin
         q_push(c, elem);
         read (elem);
   end;
End;
```

Tipos Abstractos de Datos - Ejemplo

```
Function simetrica (p: pila; c: cola): boolean;
 var
   elemP, elemC: integer;  ok: boolean;
 begin
    ok:=true;
    if (st_length(p) <> q_length(c)) then ok:=false
                                          else begin
                                               while ((ok) and (not st_empty(p)))
                                                 do begin
                                                    q_pop(c,elemC);
                                                    st_pop(p,elemP);
                                                    if (elemC <> elemP) then ok:= false;
                                          end;
    simetrica:= ok;
 end;
```