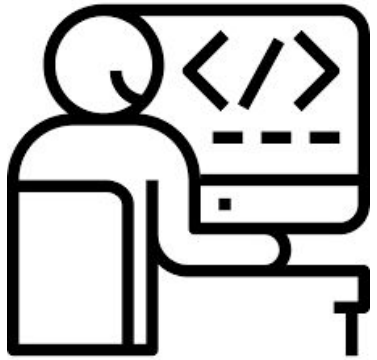


Algoritmos y Programación I

AyPI – **Temas de las clases pasadas**

- Alocación estática y alocación dinámica
- Tipo de datos puntero
- Tipo de datos lista
- Características
- Operaciones

AyPI – **Temas de la clase de hoy**

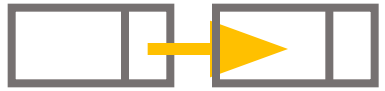
- Tipo de datos lista (Continuación)
- Operaciones (actualizar, buscar, insertar y eliminar elementos)

AyPI – **Tipo de Dato - LISTA**

Realice un módulo que reciba una lista de inmuebles y devuelva una nueva lista con los mismos inmuebles pero con sus precios aumentados en un porcentaje recibido por parámetro.

```
Program uno;
uses GenericLinkedList;

Type
        inmueble = record
                        ...
                        end;
    ListaI = specialize LinkedList<inmuebles>;

Var
    listaInmuebles: ListaI;
```

AyPI – **Tipo de Dato - LISTA**



```
function aumentarPrecio(l:listaI; porcentaje:real): listaI;
var lNueva: listaI;
    actual: inmueble;
Begin
   lNueva:= ListaI.create();
   l.reset();
   while(not l.eol()) do
      begin
      actual:= l.current();
      actual.precio:= actual.precio * (1 + porcentaje / 100);
      lNueva.add(actual);
      l.next();
   end;
   aumentarPrecio:= lNueva;
end;
```

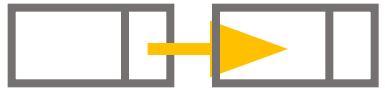
¿Cuánta memoria estamos utilizando?

AyPI – **Tipo de Dato - LISTA**

Realice un módulo que reciba una lista de inmuebles y modifique el precio de los inmuebles dentro de la misma lista. El porcentaje a aumentar es recibido por parámetro.

```
Program uno;
uses GenericLinkedList;

Type
        inmueble = record
                        ...
                   end;
    ListaI = specialize LinkedList<inmuebles>;

Var
    listaInmuebles: ListaI;
```

AyPI – **Tipo de Dato - LISTA**



```
procedure aumentarPrecio_2(l:listaI; porcentaje:real);
var  inmu: inmueble;
Begin
l.reset();
while(not l.eol()) do
   begin
   inmu:= l.current();
   inmu.precio:= inmu.precio * (1 + porcentaje / 100);
   l.setCurrent( inmu );
   l.next();
   end;
end;
```

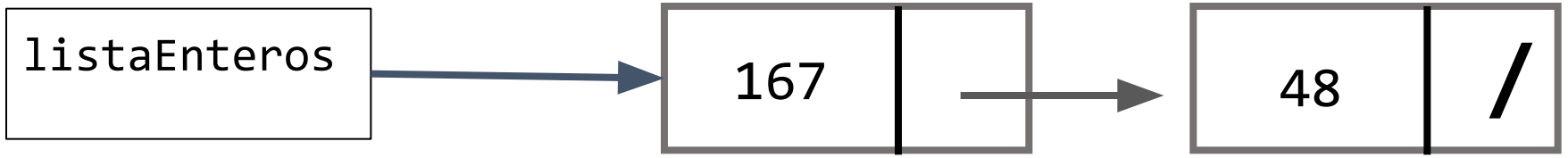
AGREGAR ADELANTE EN UNA LISTA

Implica generar un nuevo nodo y agregarlo como primer elemento de la lista.

Agrego el 48

¿Cómo lo usamos en un ejemplo?

Agrego el 167



AGREGAR ADELANTE EN UNA LISTA

Implica invocar al módulo addFirst de la lista.

```
Program uno;
uses GenericLinkedList;

Type
    ListaE = specialize LinkedList<integer>;

Var
    listaEnteros: ListaE;
Begin
  armarListaInvertida(listaEnteros);
End.
```

```
procedure armarListaInvertida(var L : listaE);
var
  i, num: integer;
begin
    L := ListaE.create();
    for i:= 1 to 15 do begin
      read(num);
      L.addFirst(num);
  end;
end;
```

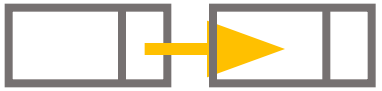
BUSCAR UN ELEMENTO

Implica recorrer la lista desde el comienzo pasando nodo a nodo hasta encontrar el elemento buscado o que se termine la lista.

```
Program uno;
uses GenericLinkedList;

Type
   ListaE = specialize LinkedList<integer>;

Var
   listaEnteros: ListaE;
```

AyPI – **Tipo de Dato - LISTA**

BUSCAR



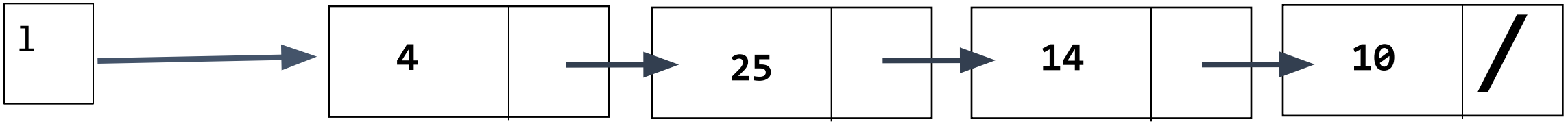
```
Program uno;
uses GenericLinkedList;

Type
    ListaE = specialize LinkedList<integer>;

Var
    listaEnteros: ListaE;
    num:integer;
    esta: boolean;



Begin
  listaEnteros := ListaE.create;
  cargarLista (listaEnteros);
  read (num);

  esta:= buscar (listaEnteros, num);
 End.
```

BUSCAR UN ELEMENTO

```
Me posiciono en el primer elemento

mientras (no sea el final de la lista y no encuentre el elemento)

    si es el elemento buscado detengo la búsqueda
    sino
      avanzo al siguiente elemento
```

AyPI – **Tipo de Dato - LISTA**

BUSCAR UN ELEMENTO



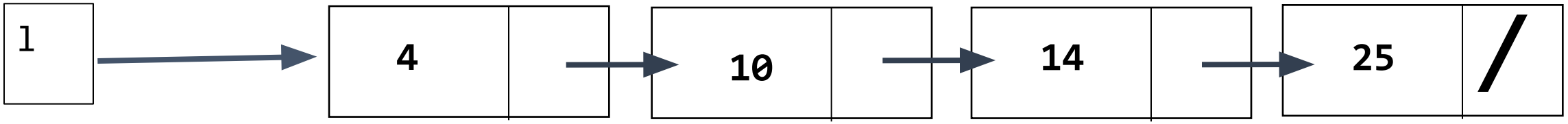

```
function buscar (l: listaE; valor: integer):boolean;
Var encontre: boolean;
Begin
  encontre:= false;
  l.reset();
  while ((not l.eol()) and (encontre = false)) do
   begin
      if (l.current() = valor) then encontre:= true
                  else l.next();
   end;
   buscar:= encontre;
end;
```

¿Funciona si la lista que recibo es vacía?
Sí

¿Funciona si la lista está ordenada?
Si funciona, ¿es la mejor solución? NO

AyPI – **Tipo de Dato - LISTA**

BUSCAR UN ELEMENTO




```
function buscarEnListaOrdenada (l: listaE; valor: integer):boolean;
Var encontre: boolean;
Begin
  encontre:= false;
  l.reset();
  while ((not l.eol()) and (l.current() < valor)) do
       l.next();

  if (not l.eol()) and (l.current() = valor) then
       encontre:= true;
  buscarEnListaOrdenada:= encontre;
end;
```

¿Funciona si la lista que recibo es vacía? Sí

¿Necesito respetar el orden de la doble condición? Sí

¿En qué difiere esta búsqueda a la búsqueda en un vector ordenado?

AyPI – **Tipo de Dato - LISTA**

INSERTAR

INSERTAR UN ELEMENTO

Se necesita que la estructura tenga un orden e implica agregar el elemento a la lista de manera que la misma siga ordenada.

```
Program uno;
uses GenericLinkedList;

Type
    ListaE = specialize LinkedList<integer>;

Var
    listaEnteros: ListaE;
```

AyPI – **Tipo de Dato - LISTA**

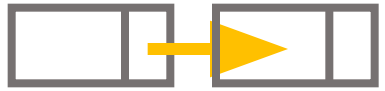
INSERTAR EN UNA LISTA

```
Program uno;
uses GenericLinkedList;

Type
    ListaE = specialize LinkedList<integer>;

Var
    listaEnteros: ListaE;

Begin
  armarListaOrdenada(listaEnteros);
End.
```

```
procedure armarListaOrdenada(var L : listaE);
var
  i, num: integer;
begin
    L := ListaE.create();
    for i:= 1 to 15 do begin
      read(num);
      insertar(L, num);
  end;
end;
```

AyPI – **Tipo de Dato - LISTA**

INSERTAR EN UNA LISTA



```
Procedure insertar (var l: listaE; valor:integer);
Var seguir: boolean;
Begin
  l.reset();
  seguir:= true;
  while(not l.eol()) and seguir do
  begin
   if l.current() > valor then
      seguir:= false
   else
      l.next();
  end;
  l.insertCurrent(valor);
end;
```

ELIMINAR UN ELEMENTO

Implica recorrer la lista desde el comienzo pasando nodo a nodo hasta encontrar el elemento y en ese momento eliminarlo (removeCurrent). El elemento puede no estar en la lista

```
Program uno;
uses GenericLinkedList;

Type
    ListaE = specialize LinkedList<integer>;

Var
    listaEnteros: ListaE;
```

AyPI – **Tipo de Dato - LISTA**

ELIMINAR EN UNA LISTA

```
Program uno;
uses GenericLinkedList;

Type
    ListaE = specialize LinkedList<integer>;

Var
    listaEnteros: ListaE;
    num: integer;
    exito : boolean;
Begin
  armarLista(listaEnteros);
  read (num);
  eliminar (listaEnteros, num, exito);
  if (exito) then
    writeln('Elemento eliminado satisfactoriamente')
  else
    writeln('No se eliminó ningún elemento');
End.
```

AyPI – **Tipo de Dato - LISTA**

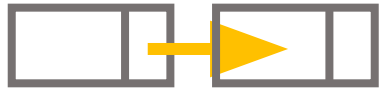
ELIMINAR EN UNA LISTA

```
Procedure eliminar(var l:listaE; valor:integer; var encontre: boolean);
begin
   l.reset();
   encontre:= false;
   while (not l.eol()) and not encontre do
      if l.current() = valor then
         encontre:= true
      else
         l.next();

   if (encontre) then
      l.removeCurrent();
end;
```

¿Funciona si la lista que recibo es vacía?
Sí, pero no hace nada

Si la lista está ordenada, ¿conviene utilizar esta solución? NO

AyPI – **Ejercicio 1**



Implemente un módulo que reciba una lista de inmuebles y una localidad. El módulo debe eliminar de la lista todos los inmuebles de dicha localidad, devolviendo la cantidad de inmuebles eliminados.

```
Procedure eliminarTodos(var l:listaI; loc:string; var cant: integer);
var exito: boolean;
begin
    cant:= 0;
    while (buscar(l, loc))  do
        begin
          eliminar(l, loc, exito);
          if exito then cant:= cant + 1;
        end;
end;
```

Analizar:

- ¿cumple la consigna?
  Sí
- ¿es una solución eficiente en tiempo de ejecución?
  No

¿cómo lo mejoramos?

AyPI – **Ejercicio 1**

Implemente un módulo que reciba una lista de inmuebles y una localidad. El módulo debe eliminar de la lista todos los inmuebles de dicha localidad, devolviendo la cantidad de inmuebles eliminados.

```
Procedure eliminarTodos (var l:listaI; loc :string; var cantEli: integer);
begin
   l.reset();
   cantEli := 0;
   while (not l.eol()) do
      if (l.current().localidad = loc) then
      begin
         l.removeCurrent();
         cantEli:= cantEli + 1;
      end
      else  l.next();
end;
```

¿Qué sucede si avanzo siempre (sin el else)? ¡Salteo elementos!

AyPI – **Ejercicio 2**

Implemente un módulo que reciba una lista de inmuebles (ORDENADA por localidad) y una localidad. El módulo debe eliminar de la lista todos los inmuebles de dicha localidad y devolver la cantidad de inmuebles eliminados.

AyPI – **Ejercicio 2**

Implemente un módulo que reciba una lista de inmuebles (ORDENADA por localidad) y una localidad. El módulo debe eliminar de la lista todos los inmuebles de dicha localidad y devolver la cantidad de inmuebles eliminados.

```
Procedure eliminarTodos (var l:listaI; loc :string; var cantEli: integer);
begin
   l.reset();  cantEli := 0;
   while (not l.eol() and (l.current().localidad < loc) do
       l.next();
   while (not l.eol()) and (l.current().localidad = loc) do begin
         l.removeCurrent();
         cantEli:= cantEli + 1;
   end;
end;
```