TEORIA

13

TEMAS de la CLASE

1 Tipo de dato LISTA ENLAZADA

- Concepto y Características
- Declaración del tipo en Pascal
- Operaciones frecuentes
- Ejercitación
- Análisis Comparativo Vector vs Lista

Listas

El concepto de una lista es bastante intuitivo ya que encontramos varios ejemplos en la vida cotidiana:

- Lista de pasajeros
- Lista de compras
- Lista de alumnos

LISTA COMPRA

Microsoft Excel - Libro1

	A	B	C	D
1	ANDRADE GONZALEZ ANA ALICIA			
2	ALMENDRA MENDEZ OSCAR DAVID			
3	CASTILLO VEGA JULIO CESAR			
4	BANDA GARCIA BRENDA			
5	CORZO MORENO LUIS ALBERTO			
6	DIAZ NUÑEZ ISRAEL			
7	CRUZ SANTOS CLAUDIA YOMELI			
8	FLORES ESCOBAR NAHUM			
9	GARCIA COLORADO MARTHA IZTAYANA			
10	HERNANDEZ REYES LUCERO MONTSERRAT			
11	HERNANDEZ ESPEJO KAREN			
12	HERNANDEZ DAVILA RAFAEL			
13	LAGUNES MENDOZA DIANA LAURA			
14	LARA RODRIGUEZ TANIA GUADALUPE			
15	LAGUNES CRUZ XOCHITL JEREMY			
16	LARA SALAS CARLOS ALBERTO			
17	MORALES SILVA RUTH EUNICE			
18	SAGUAYA REYES ANA BARBARA			
19	SOLIS PEREZ ZURISADAI			
20	SALDIVAR ARIAS SANDRA IVONNE			
21	SOTO PORTILLO DENISSE			
22	USCANGA CRUZ IVAN			
23	VELAZQUEZ ZARATE CLAUDIA GIOVANNA			
24	CABRERA MARTINEZ JOSE LUIS			
25	COLIN ROMAN ADILENE VIANEY			
26	REYES CAMPOS ELIZABETH			
27	RODRIGUEZ FUENTES ANA BELEM			
28	SILVA ORTEGA ZELTZIN NAYELI			
29	VASQUEZ SANCHEZ MARIANA			
30	WILGGIN JIMENEZ OMAR			
31				

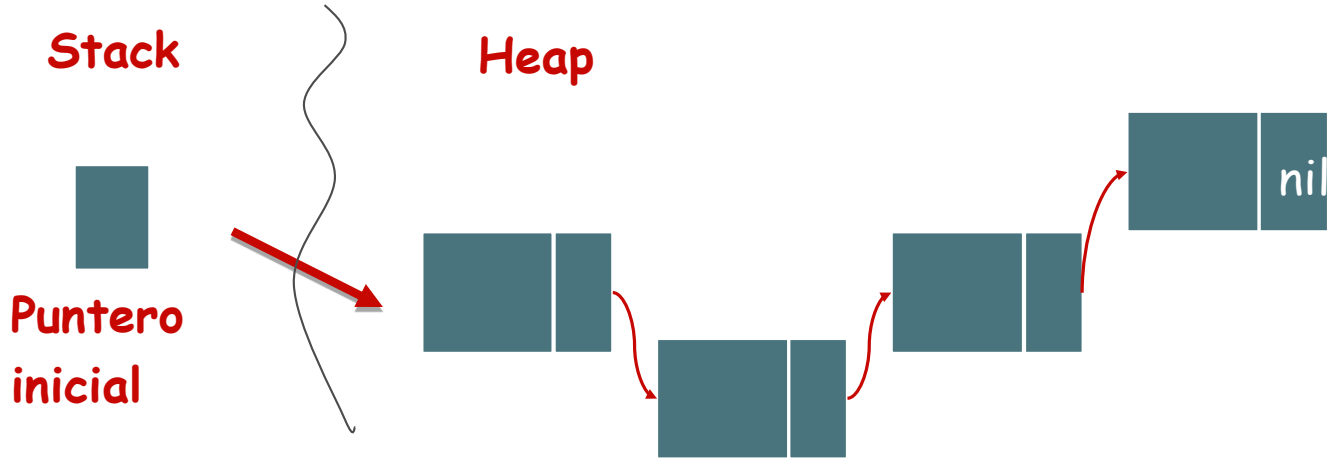
➡ Pensemos una posible representación para la estructura de datos, partir de lo que conocemos…

➡ ¿Podemos representar la información de manera mas adecuada si no conocemos cuántos elementos va a contener la estructura?

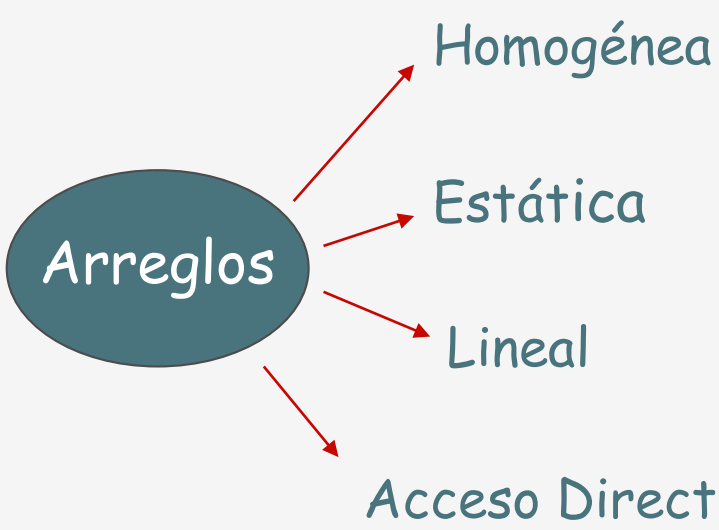
Listas - Concepto

- Colección de elementos homogéneos, con una relación lineal que los vincula, es decir que cada elemento tiene un único predecesor (excepto el primero), y un único sucesor (excepto el último).

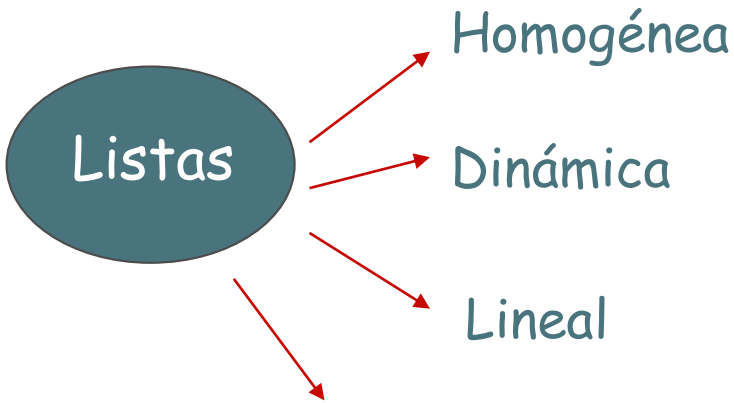
- Los elementos que la componen no ocupan posiciones secuenciales o contiguas de memoria. Es decir pueden aparecer dispersos en la memoria, pero mantienen un orden lógico interno.

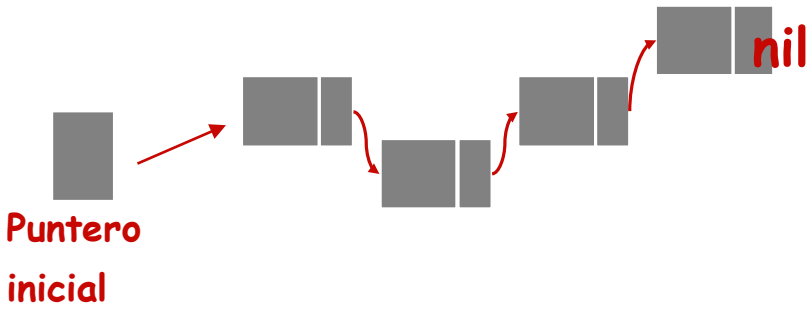


Análisis comparativo Operaciones (Listas – Arreglos)



Productos





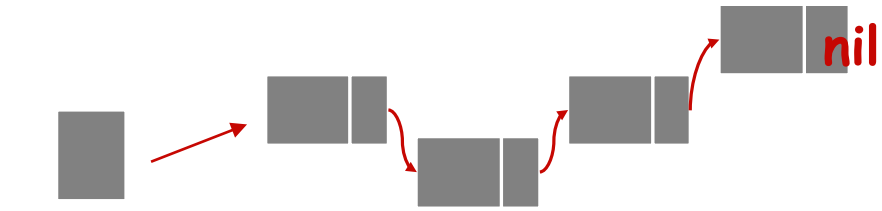
¿Cómo y dónde se almacenan los datos en cada una de las estructuras?

Análisis comparativo Operaciones (Listas – Arreglos)

¿Cómo y dónde se almacenan los datos?

Productos

- Los arreglos se almacenan en memoria estática.
- La ocupación de memoria se resuelve en tiempo de compilación.
- Ocupan posiciones consecutivas de memoria a partir de la posición inicial.



Puntero inicial

- Las listas se almacenan en memoria dinámica.
- La ocupación de memoria se resuelve en tiempo de ejecución.
- Se disponen aleatoriamente en memoria. Se relacionan lógicamente.

Análisis comparativo Operaciones (Listas - Arreglos)

¿Qué diferencias encontramos para acceder al k-ésimo elemento en un vector y en una lista?

K=2

Arreglos

Acceso Directo

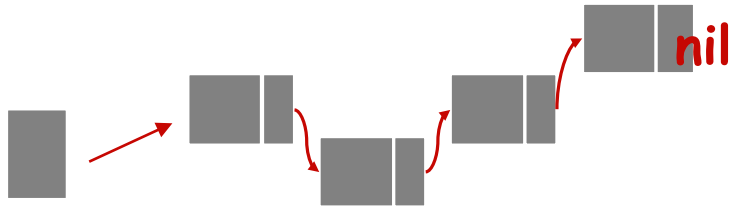
Productos

Se resuelve simplemente:

Productos [2]

Listas

Acceso Secuencial



Puntero inicial

Se debe recorrer desde el principio hasta llegar a la posición K.

Listas – Características

- Están compuesta por nodos
- Los nodos se conectan por medio de enlaces o punteros
- Cuando se necesitan agregar nodos a la estructura, se solicita espacio adicional
- Cuando existen nodos que ya no se necesitan, pueden ser borrados, liberando memoria
- Siempre se debe conocer la dirección del primer nodo de la lista (puntero inicial) para acceder a la información de la misma
- El último nodo de la lista se caracteriza por tener su enlace en Nil

Listas – Declaración en Pascal

Ejemplo de la declaración de una lista de enteros en Pascal

```
Program Ej1;
Type
    Lista= ^Nodo;
    Nodo= Record
            Datos: integer; {contenido}
            sig : Lista; {dirección del siguiente nodo}
          End;
Var
   L: Lista; {Memoria estática reservada para el puntero inicial}
```

Listas – Declaración en Pascal

En general:

```
Type
  info = …;
  Lista = ^ nodo;
  nodo = record
           Datos : info;
           Sig: Lista;
         End;

Var
  P : Lista;
```

Observaciones…

Listas – Operaciones mas frecuentes

- Recorrer una lista
- Buscar un elemento en la lista
- Crear una lista vacía
- Agregar un elemento al principio de una lista
- Agregar un elemento al final de una lista
- Eliminar un elemento de la lista
- Insertar un nuevo elemento en una lista ordenada

Recorrido de una Lista Enlazada

Se quiere imprimir los datos guardados en una lista enlazada. Para ello es necesario recorrer la lista completa, desde el primer nodo al último.

Supongamos la declaración:

```
Type
  cadena50 = string[50];
  persona= record
              nom: cadena50;
              edad : integer
           end;
  lista= ^nodo;
  nodo = record
           datos : persona;
           sig   : lista;
         end ;
```

Observar parámetro...

```
Procedure recorrido ( pri : lista);

Begin
 while (pri <> NIL) do begin
   write (pri^.datos.nom,
          pri^.datos.edad) ;
   pri:= pri^.sig
 end;
end;
```

Recorre hasta el final

Búsqueda de un elemento en una Lista

Se quiere saber si existe un elemento determinado en una lista. Se debe recorrer la lista desde el primer nodo hasta encontrar el elemento o bien hasta llegar al final de la lista.

Supongamos la declaración:

```
Type
 cadena50 = string[50];
  persona= record
             nom:cadena50;
             edad:integer
            end;
 lista = ^nodo;
 nodo = record
         datos : persona;
         sig   : lista;
        end;
```

```
function buscar (pri: lista;
                 x:cadena50):boolean;
Var
   encontre : boolean;

begin
 encontre := false;
 while (not encontre) and (pri <> NIL) do
   if x = pri^.datos.nom  then
                          encontre:= true
                           else
                          pri:= pri^.sig;
   buscar := encontre
End;
```

Recorre hasta encontrarlo

Recorre hasta el final

Crear una Lista vacía

La operación de Crear Lista Vacía es simplemente asignarle Nil a su puntero inicial. Por ejemplo:

```
Type
 cadena50 = string[50];
  persona= record
              nom:cadena50;
              edad : integer
            end;
 lista = ^nodo;
 nodo = record
          datos : persona;
          sig   : lista;
        end;
```

```
Begin

  ......
  L:=nil;
  .....

End.
```

Observar que NO se usa NEW!!

Agregar un elemento al principio de la lista

Supongamos que se ingresan el nombre y la edad de personas, hasta que se ingresa la edad 0. Los datos de cada persona se deben guardar en una lista. Nota: agregar siempre al principio de una lista.

```
Type
 cadena50=string[50];
 persona= record
            nom:cadena50;
            edad:integer;
          end;
 lista = ^nodo;
 nodo = record
          datos: persona;
          sig: lista;
        end;
Var
 L : Lista;
 p : persona;
```

```
Procedure AgregarAdelante
(var L:lista; per:persona);

Var nue:Lista;
Begin
  New(nue);
  nue^.datos:=per;
  nue^.sig:=L;
  L:=nue;
End;
```

```
Begin {prog. ppal}
 L:=nil;
 leerPersona (p);
 While (p.edad<>0) do Begin
    AgregarAdelante (L, p);
    leerPersona (p);
 End;
End.
```

AgregarAdelante recibe como parámetros el puntero inicial de la lista y los datos de lal persona que se guarda

Ejercitación I

Se desean procesar los productos de una venta del supermercado. De cada producto se conoce código, nombre, marca, precio y fecha de vencimiento. El ingreso de los productos finaliza cuando se lee el código -1. Se pide generar una lista con los códigos e identificación de los productos de marca "SanCor".

100
Bebida
Coca Cola
10
31-12-2017

121
Nido
Leche en polvo
35
30-01-2020

80
Leche Larga Vida
SanCor
18
30-01-2020

130
Amanda
Yerba
25
31-12-2015

50
Arcor
Tomate al natural
15
25-10-2016

75
Ilolay
Leche Larga vida
25
31-12-2016

200
Dulce de leche
SanCor
15
31-12-2016

Agregar un elemento al final de la lista

Supongamos que se ingresan el nombre y la edad de personas, hasta que se ingresa la edad 0. Los datos de cada persona se deben guardar en una lista, respetando el orden de ingreso.

```
Type
 cadena50 = string[50];
 persona= record
            nom:cadena50;
            edad:integer
           end;
 lista = ^nodo;
 nodo = record
          datos: persona;
          sig : lista;
         end;

Var
 L : Lista;
 p : persona;
```

```
Procedure AgregarAlFinal
(var L:lista; per:persona);

Var
   ……..
Begin
    …..;
End;
```

```
Begin {prog. ppal}
 L:=nil;
 leerPersona (p);
 While (p.edad<>0) do Begin
   AgregarAlFinal (L, p);
   leerPersona (p);
 End;
End.
```

AgregarAlFinal recibe como parámetros el puntero inicial de la lista y los datos de la persona que se guarda

Agregar un elemento al final de la lista

```
procedure AgregarAlFinal (var pri: lista; per: persona);
var  act, nue : lista;

begin
 new (nue);
 nue^.datos:= per;
 nue^.sig := NIL;
 if pri <> Nil then begin
                      act := pri ;
                      while  (act^.sig <> NIL ) do act := act^.sig ;
                      act^.sig := nue ;
                    end
                    else
                      pri:= nue;
end;
```

¿Se puede mejorar el tiempo de ejecución del proceso?

¿Qué modificaciones se deben hacer?

Agregar un elemento al final de la lista (otra solución)

Podríamos plantear la operación de AgregarAlFinal2 como un procedimiento que recibe el puntero inicial, el puntero al último nodo y el número a guardar…

```
Type
 cadena50 = string[50];
 persona= record
           nom: cadena50;
           edad : integer
          end;
 lista = ^nodo;
 nodo = record
         datos: persona;
         sig: lista;
        end;
Var
 L, Ult : Lista;
 p : persona;
```

```
Procedure AgregarAlFinal2
(var L, Ult: lista; per:persona);

Var
 ….
Begin
 ….
 …..
End;
```

```
Begin {prog. ppal}
 L:=nil; ult:=nil;
 leerPersona (p);
 While (p.edad<>0) do begin
   AgregarAlFinal2 (L, Ult, p);
   leerPersona (p);
 End;
End.
```

Agregar un elemento al final de la lista (otra solución) (con puntero al último nodo)

```
procedure AgregarAlFinal2 (var pri, ult: lista; per: persona);
var  nue : lista;

begin
 new (nue);
 nue^.datos:= per;
 nue^.sig := NIL;
 if pri <> Nil then
                  ult^.sig := nue;
                else
                  pri := nue;
 ult := nue;
end;
```

Si la lista tiene elementos

Si la lista no tiene elementos

Borrar un elemento de la lista

Supongamos que se dispone de una lista de personas y se quiere eliminar a una persona cuyo nombre se lee de teclado, de ser posible.

```
Type
 cadena50 = string[50];
 persona= record
           nom: cadena50;
           edad: integer;
          end;
 lista = ^nodo;
 nodo = record
         datos: persona;
         sig: lista;
        end;
Var
 L : Lista;
 nombre : cadena50;
 éxito: boolean;
```

```
Procedure BorrarElemento
(var L:lista; nom:cadena50);

Var
   ……..
Begin
   …..;
   ….
End;
```

```
Begin {prog. ppal}
 CargarLista (L);
 read (nombre);
 BorrarElemento(L, nombre, exito);
 if éxito then write ('Se eliminó')
          Else write ('No existe');
End.
```

AgregarAlFinal recibe como parámetros el puntero inicial de la lista y los datos de lal persona que se guarda

Borrar un elemento determinado de la lista

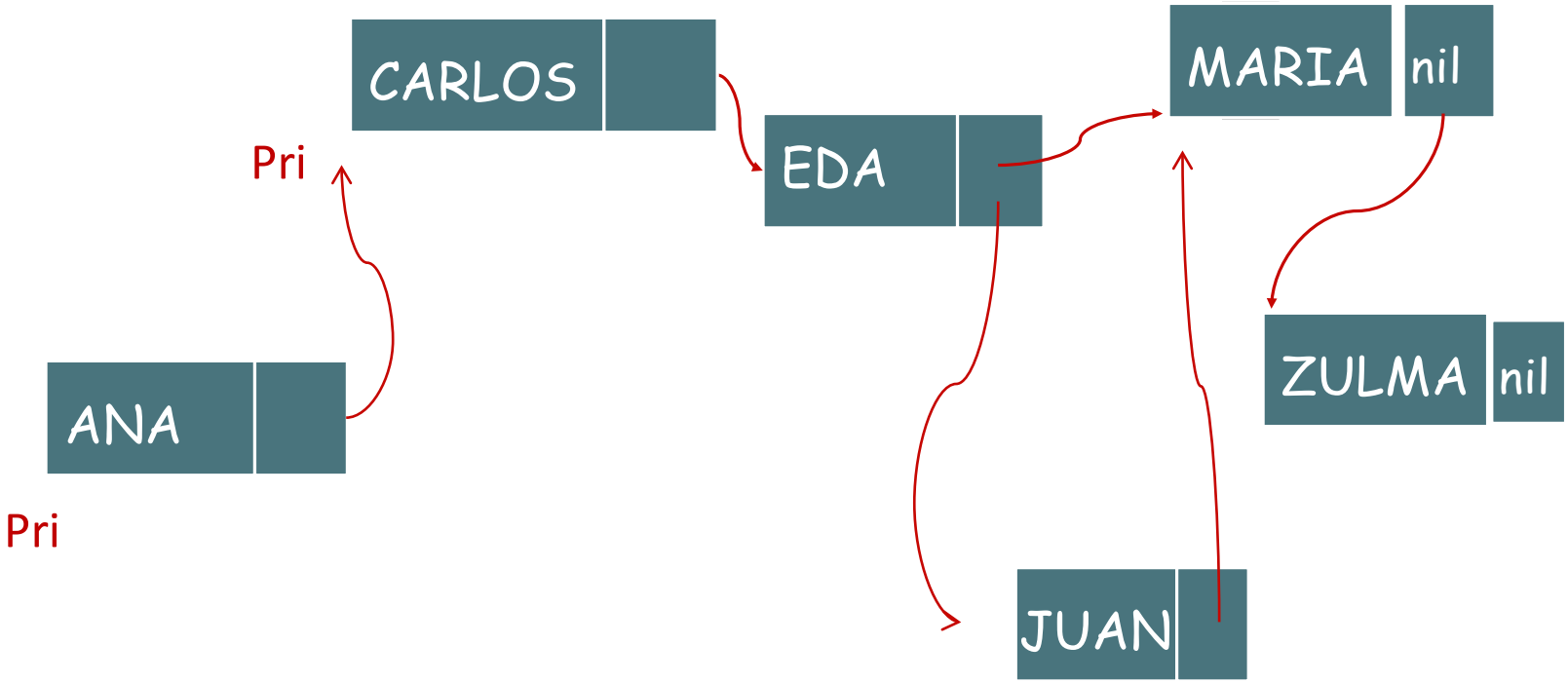
```
Procedure BorrarElemento (var pri:lista; nom:cadena50;
                          var exito: boolean);
var ant, act: lista;
begin
  exito := false;
  act := pri;
  ant := pri;
 {Recorro mientras no se termine la lista y no encuentre el elemento}
  while  (act <> NIL)  and (act^.datos.nom <> nom) do begin
      ant := act;
      act := act^.sig
  end;
  if (act <> NIL) then begin
                    exito := true;
                     if (act = pri) then  pri := act^.sig;
                                    else  ant^.sig:= act^.sig;
                     dispose (act);
                  end;
 end;
```

El dato a borrar es el primero

El dato a borrar es uno cualquiera

Insertar un elemento en una lista

Supongamos que tenemos una lista de personas ordenadas alfabéticamente y queremos insertar los nombres Ana, Zulma y Juan.



Insertar un elemento en una lista

Pasos a seguir:

1. Pedir espacio en memoria para el nuevo nodo
2. Guardar los datos en el nodo
3. Buscar posición donde se debe insertar (secuencialmente a partir del puntero inicial)
4. Reacomodar punteros. Considerar los tres casos:
    a. El nuevo elemento va en el inicio de la lista.
    b. El nuevo elemento va en el medio de dos existentes.
    c. El nuevo elemento va al final de la lista.

Insertar un elemento en una lista

Supongamos que se dispone de una lista de personas ordenada alfabéticamente por el nombre y se desea incorporar la información de una persona a dicha lista. Los datos de la persona se leen de teclado.

```
Type
 cadena50 = string[50];
 persona= record
            nom:cadena50;
            edad:integer;
          end;
 lista = ^nodo;
 nodo = record
          datos:persona;
          sig:lista;
        end;
Var
 L: Lista;
 p: persona;
```

```
Procedure InsertarElemento
(var L:lista; per:persona);

Var
   ……..
Begin
   …..;
   ….
End;
```

```
Begin {prog. ppal}
  CargarLista (L);
  leerPersona (p);
  InsertarElemento (L, p);
End.
```

InsertarElemento recibe como parámetros el puntero inicial de la lista y los datos de la persona que se guarda

Insertar un elemento en una lista

```
Procedure InsertarElemento ( var pri: lista; per: persona);
var ant, nue, act: lista;
begin
  new (nue);
  nue^.datos := per;
  act := pri;
  ant := pri;
{Recorro mientras no se termine la lista y no encuentro la posición
correcta}
  while (act<>NIL) and (act^.datos.nombre < per.nombre) do begin
      ant := act;
      act := act^.sig ;
  end;
  if (ant = act)  then pri := nue   {el dato va al principio}
                  else  ant^.sig  := nue; {va entre otros dos o al final}
  nue^.sig := act ;
end;
```

Ejercitación II

Se desea simular la facturación de una compra de supermercado. De cada producto se conoce código, identificación, marca, precio y fecha de vencimiento. El ingreso de los productos finaliza cuando se lee el código -1. Se pide imprimir el ticket de compra con: cantidad total de productos vendidos, monto total de la compra y el detalle de los productos comprados (Nombre y precio).

Análisis Comparativo Vector vs Lista Enlazada

Analicemos algunas consideraciones que entran en juego en el momento de elegir una estructura de las vistas hasta ahora:

- **Espacio**
- **Tiempo**
  - Recuperar datos
  - Almacenar datos
- **Parámetros**

Análisis Comparativo - Espacio

Espacio: se refiere a la cantidad de memoria consumida por una estructura de datos dada.

Si se supone igual cantidad de datos en las dos estructuras se puede afirmar que:

- Vectores : son más económicos.
- Listas : requieren espacio extra para los enlaces.

Si no se conoce la cantidad de datos que contendrá cada estructura, que otro análisis se puede hacer en:

- Vectores?
- Listas?

Análisis Comparativo - Tiempo

Tiempo: se refiere al tiempo que toma almacenar o recuperar datos, entre otros.

Se tendrá que analizar:

- **Tiempo empleado para Recuperar Datos en:**
  - Vectores
  - Listas
- **Tiempo empleado para Almacenar Datos en:**
  - Vectores
  - Listas

Análisis Comparativo

Parámetros: analizar cual es el costo del pasaje de parámetros por valor y por referencia en:

- Arreglos
- Listas