Programación II

Profesor: Alejandro H. González

JTP: Sabrina Martorelli

Ayudantes: María Lucía VIOLINI, Leandro Matías ROMANUT, Gonzalo GARCIA CHICOTE, César ESTREBOU

Colaboradores: Mariano Sánchez, Julieta Corvi

Horarios de teoría: Lunes y miércoles de 8 a 10 hs. Aula 10B

Horarios de práctica: Lunes 10:00 a 12:00 Aula 10B

Miércoles 10:00 a 12:00 Aula 10B

Programación II -Aprobación de la asignatura

1 parcial practico con dos recuperatorios. Aprobado

Opción 1 Régimen de Promoción

1. Aprobar dos evaluaciones cortas sobre la Teoría que se tomarán en las fechas estipuladas en el cronograma. (durante el curso)

2. Aprobar el parcial práctico en la primera o segunda fecha.

3. Cumplidas las 2 primeras condiciones, podrán rendir un parcial teórico en la fecha del 2do. recuperatorio de parcial práctico. Si el alumno aprueba este parcial teórico con nota mayor de 6 (seis) tendrá aprobada la promoción.

Programación II -Aprobación de la asignatura

Los alumnos que tengan aprobada la promoción deberán inscribirse y presentarse en la siguiente mesa de exámenes finales, para que se registre oficialmente la nota del examen.

Programación II -Aprobación de la asignatura

Opción 2 Con examen final

Los alumnos que no hayan aprobado el régimen de promoción, pero hayan aprobado el parcial práctico, obtendrán la aprobación de los Trabajos Prácticos y la habilitación para rendir el Examen Final de la asignatura.

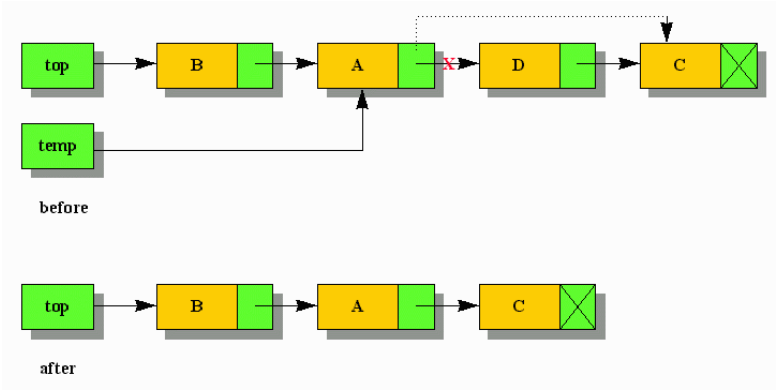
El examen final es de carácter teórico-práctico

Programación II –en WebUNLP

Deben solicitar inscripción en el curso Programación II, donde el profesor es Alejandro González.

Listas Repaso

Ejemplos



Número	Título	Artista
1	The Sign	Ace of Base
2	A Quién Le Importa	Alaska y Dinarama
3	Corazón Partío	Alejandro Sanz
4	No One	Alicia Keys
5	Flaca	Andres Calamaro
6	Complicated	Avril Lavigne
7	Baila Mi Corazón	Belanova
8	Stand By Me	Ben E. King
9	Irreplaceable	Beyoncé
10	Completamente	Chetes
11	Yellow	Coldplay
12	Quién Me Iba A Decir	David Bisbal
13	Personal Jesus	Depeche Mode
14	Cada Dos Minutos	Despistaos
15	White Flag	Dido
16	Mercy	Duffy
17	En Algun Lugar	Duncan Dhu
18	Let's Groove	Earth, Wind & Fire
19	Demasiado Tarde	El Sueño de Morfeo
20	No Te Pido Flores	Fanny Lu
21	Che Che Cole	Héctor Lavoe
22	Olé	Jarabe de Palo
23	Ring Of Fire	Johnny Cash
24	Love At First Sight	Kylie Minogue
25	Perdoname	La Factoria
26	Bleeding Love	Leona Lewis
27	Un Hombre Busca Una Mujer	Luis Miguel
28	Makes Me Wonder	Maroon 5
29	Rome Wasn't Built In A Day	Morcheeba
30	In Bloom	Nirvana
31	Young Folks	Peter Bjorn And John
32	Olvidame Y Pega La Vuelta	Pimpinela
33	Another One Bites The Dust	Queen
34	Fake Plastic Trees	Radiohead
35	Umbrella	Rihanna
36	Listen To Your Heart	Roxette
37	Love Song	Sara Bareilles
38	Bidi Bidi Bom Bom	Selena
39	A Puro Dolor	Son by Four
40	ABC	The Jackson 5

LISTAS- CONCEPTOS

Una lista es 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.

LISTAS - CARACTERISTICAS

✓Se crean a partir de punteros.
✓Son estructuras donde se almacenan datos sin saber la cantidad de los mismos.
✓Es una estructura dinámica: se reserva/libera memoria para datos según sea conveniente.
✓Siempre debo guardar el puntero inicial de la lista, es decir el apuntador al primer nodo, para luego poder recorrerla, ya que a partir del primer elemento se puede acceder al siguiente y así sucesivamente. acceso secuencial.

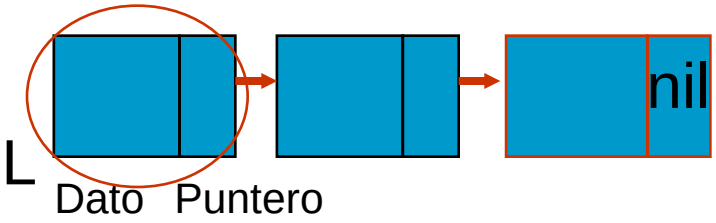
LISTAS - Características

Está compuesta por nodos.

Los nodos se conectan por medio de enlaces o punteros.

Cuando se necesita espacio adicional, nuevos nodos pueden ser alocados y agregados a la estructura (New).

Cuando existen nodos que ya no se necesitan, pueden ser borrados, liberando memoria (Dispose).



Cómo se declara la estructura?

LISTAS – Declaración del Tipo

```
Program uno;

Type nombreTipo= ^nombreElemento;

     nombreElemento= Record

                       elementos : tipoElementos;

                       punteroSig: nombreTipo;

                     End;

Var L: nombreTipo; {Memoria estática reservada}
```

Debe declararse primero

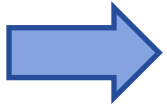
Cualquier tipo de los vistos

Estructura recursiva

Lista de enteros?

LISTAS – Declaración del Tipo

```
Program uno;
Type Lista= ^Nodo;
        Nodo= Record
                    Datos: integer;
                    Pun: Lista;
                End;
Var L: Lista; {Memoria estática reservada}
```



Cómo trabaja la memoria?

LISTAS - MEMORIA

```
Type
    Alumno= record
                    Nombre : string [20];
                    dni : string[10];
                   end;
   Lista= ^ nodo;
   nodo =  record
              datos: alumno;
              sig:lista;
             end;

  lis : lista;
```

Memoria= 4

LISTAS - OPERACIONES

- ✓ Crear lista agregando los elementos al inicio.
- ✓ Crear lista agregando los elementos al final
- ✓ Insertar un nuevo elemento en una lista ordenada
- ✓ Recorrer una lista
- ✓ Acceder al k-ésimo elemento de la lista
- ✓ Eliminar un elemento de la lista
- ✓ Combinar dos listas ordenadas formando una sola ordenada (Merge de Listas)

LISTAS – AGREGAR AL INICIO

```
Var L: Lista;
       n:integer;
Begin
  L:= nil;
  read (n);
  while (n <> 48) do
   begin
      agregar (L,n);
      read (n);
   end;
End.
```

Indica que la lista no tiene elementos

LISTAS – AGREGAR AL INICIO

```
Procedure agregar (Var pri: Lista; n:integer);
var
    nuevo:lista;
Begin
 new (nuevo);     nuevo^.sig:= nil;      nuevo^.datos:=n;

  if (pri = nil) then  pri:= nuevo
  else begin
          nuevo^.sig:= pri;
          pri:= nuevo;
    end;
End.
```

Creo un nuevo nodo

Determino si es el primer elemento

Actualizo el primer elemento

Se puede mejorar

LISTAS – AGREGAR AL INICIO

El algoritmo anterior se puede mejorar.

¿Que diferencias se ven a cuando queremos insertar al principio en un vector?

LISTAS – RECORRER LA LISTA

Suponga que se tiene creada una lista. Se pide realizar un módulo que imprima los elementos de la lista

```
Var L: Lista;
      n:integer;
Begin
  L:= nil;
  crearLista (L);
  imprimirLista (L);
End.
```

LISTAS – RECORRER LISTA

```
Procedure imprimirLista (pri:
Lista);

Begin
  while (pri <> nil) do
      begin
         write (pri^.datos);
         pri:= pri^.sig;
      end;
End.
```

1- ¿Por qué paso la lista por valor?

2- ¿Por qué en la condición del while no escribo (pri^.sig <> nil)?

LISTAS – ELIMINAR VALOR

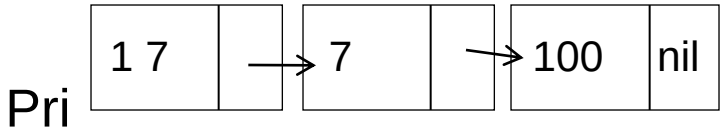
Suponga que se tiene creada una lista. Se pide realizar un módulo que lea un número y elimine dicho número de la lista

```
Var L: Lista;
      n:integer;
Begin
  L:= nil;
  crearLista (L);  -- no se implementa
  read (n);
  eliminarValor (L,n);
End.
```

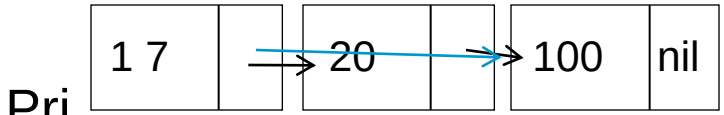
¿Cuántos casos debo considerar?

LISTAS – ELIMINAR VALOR

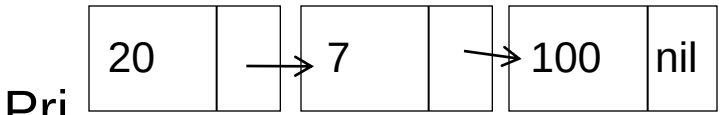
n=20



El elemento no está



El elemento está, pero no al principio



El elemento está, al principio

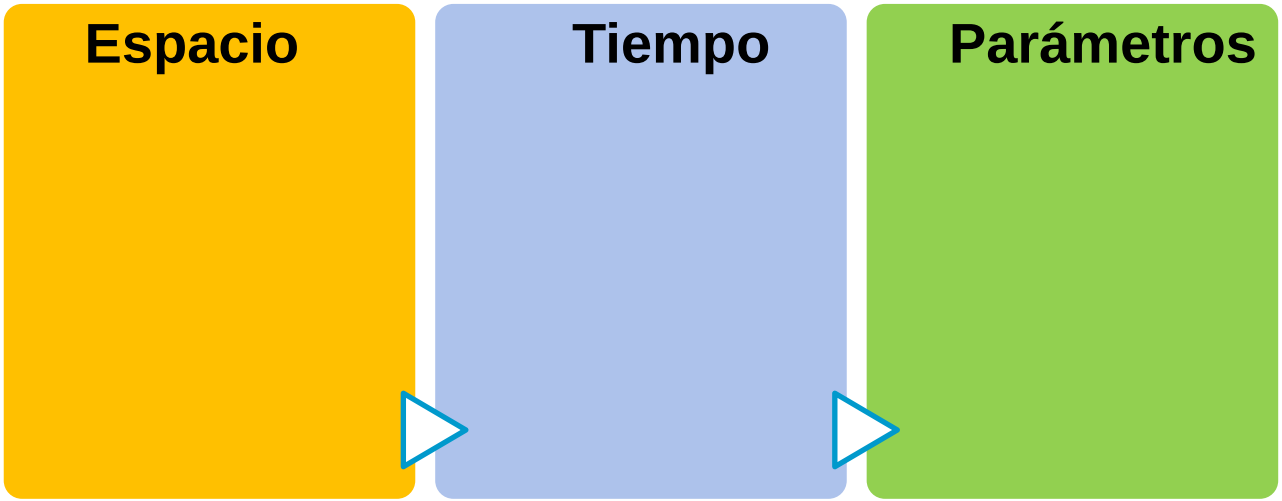
LISTAS – ELIMINAR VALOR

```
Procedure eliminarValor (var pri: Lista;
n:integer);
Var pos,ant:lista;
Begin
  pos:= pri; ant:= pri; ok:= false;
  while (pos <> nil) and (not ok)do
      if (pos^.datos = n) then ok:= true
      else begin
          ant:=pos;
          pos:= pos^.sig;
       end;
  if (ok=true) then
   begin
      if (pos = pri) then
          pri:= pos^.sig
      else begin
         ant^.sig:= pos^.sig
      dispose (pos);
   end;
End;
```

Modificar el algoritmo suponiendo que puede aparecer mas de una vez

LISTAS – CONCLUSIONES

Analicemos qué factores entran en juego al elegir cualquiera de las estructuras conocidas hasta el momento:



LISTAS CONSIDERACIONES

Espacio: se refiere a la cantidad de memoria utilizada por la estructura de datos.

Suponiendo que tienen la misma cantidad de datos

- los arreglos ocupan menos memoria que las listas (por los enlaces). En qué caso teniendo la misma cantidad de elementos el arreglo ocupa mas?

LISTAS CONSIDERACIONES

Tiempo: se refiere al tiempo que toma recuperar un dato de la estructura.

- acceso secuencial utilizado por las listas, para acceder al 4to elemento debo pasar por los tres anteriores.

- acceso random utiliza el arreglo, para acceder a al 4to elemento a un desplazamiento base le suma la cantidad de elementos que se tiene que mover * tamaño del elemento

  Dirección Elemento Actual = Dirección Base + (Offset * Indice)

LISTAS CONSIDERACIONES

Tiempo: se refiere al tiempo que toma recuperar un dato de la estructura.

- acceso secuencial requiere un tiempo que no es constante, depende el número del elemento al que se quiere acceder.

- acceso random requiere un tiempo fijo

LISTAS CONSIDERACIONES

Datos: se refiere a como se almacenan los datos.

- Listas: siempre hay lugar, distintos casos para agregar al principio, final u ordenado.
- Arreglos: no siempre hay lugar, distintos casos para agregar al principio, final u ordenado.

LISTAS CONSIDERACIONES

Parámetros: por valor o por referencia

- Valor: de la lista se copia solo el puntero inicial, del arreglo se copia todo.
- Referencia: de la lista se copia el valor inicial para poder ser modificado, del arreglo se copia la dirección en donde se encuentra almacenado.

LISTAS CIRCULARES

Estructura de Datos LISTA CIRCULAR - MOTIVACION

Juego de la Oración

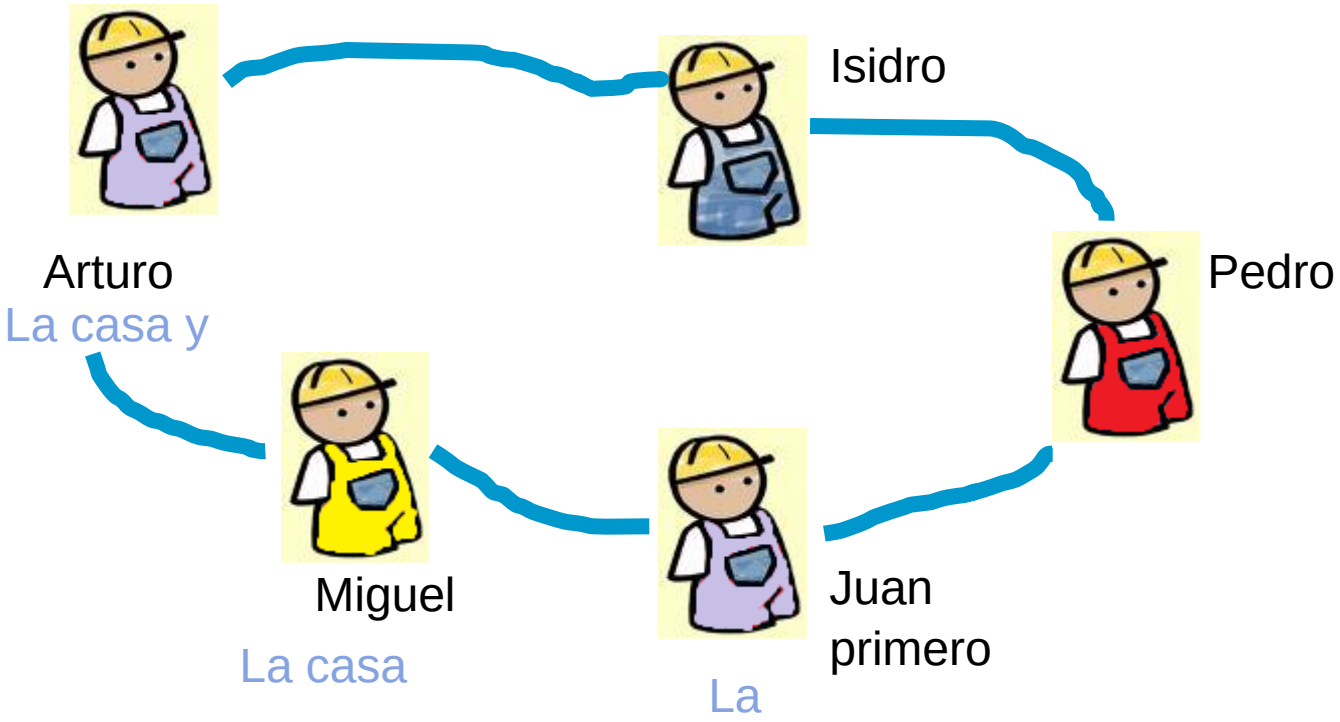
Supongamos un grupo de amigos que se juntan a jugar al juego de la oración.

Se disponen los participante en una ronda. Cada persona debe ir agregando una palabra, a una oración que se va creando conjuntamente.

Es decir, se inicia por un participante diciendo una palabra, el segundo debe decir la palabra que dijo el anterior más una nueva, el tercero las dos palabras anteriores más una nueva, etc.

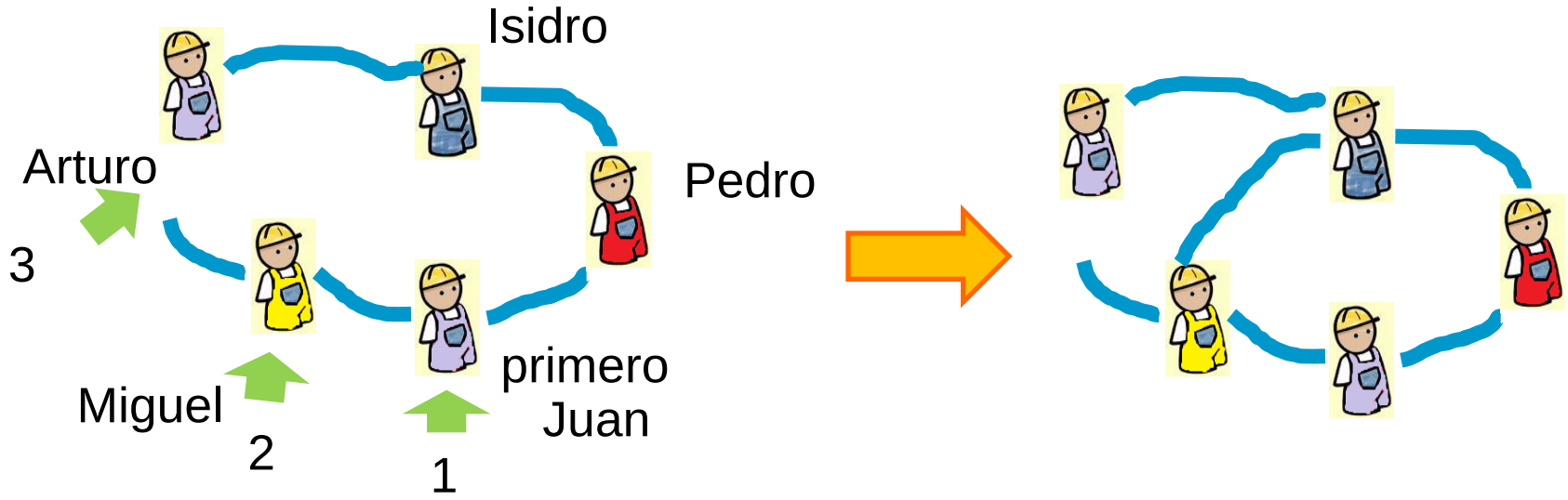
Así si algún participante olvida la oración o agrega más de una palabra queda eliminado.

Estructura de Datos LISTA CIRCULAR-MOTIVACION



Estructura de Datos LISTA CIRCULAR - MOTIVACION

Supongamos que Arturo pierde entonces debe salir de la ronda



Ahora el juego continúa a partir del siguiente jugador respecto del que perdió anteriormente

Estructura de Datos LISTA CIRCULAR - MOTIVACION

Así continuará el juego hasta lograr dejar una sola persona en la ronda, la cual es la ganadora

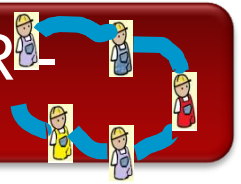
¿Qué ocurre si utilizamos una lista simplemente enlazada para representar esta situación?

¿Qué ocurre con el recorrido en ese caso?

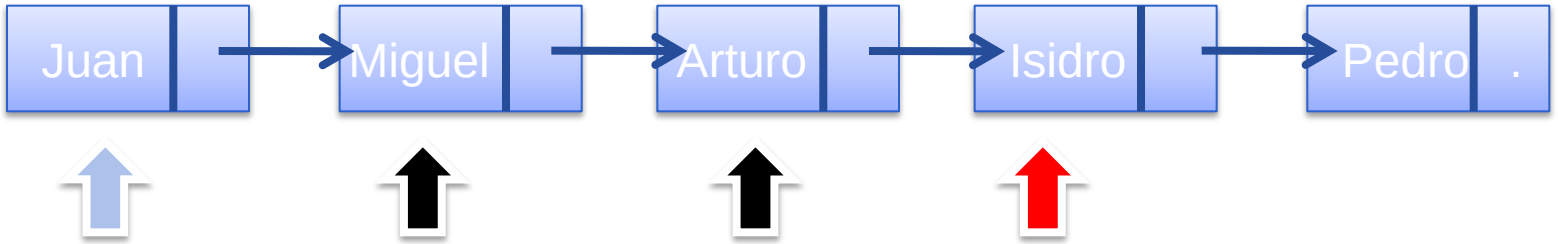
JUEGO DE LA ORACION

- ✓ Solución con LISTAS SIMPLES

Estructura de Datos LISTA CIRCULAR - MOTIVACION



Supongamos que la estructura es una lista simple. ¿Qué ocurre?



```
procedure juegoOración( var pri : juego; var ganador : niño);
var     ult, p : juego; long: integer; frase, fraseAnterior: string;
Begin
 p := pri; long:= longitudLista(pri);
 while ( long > 1) do
    begin
      fraseAnterior:=´ ´;  readln(frase);
      while (EsCorrecta (frase, fraseAnterior) = true)  do
          begin
                ult:= p;
                 if (p^.sig = Nil) then p := pri
                                       else p := p^.sig;
                fraseAnterior.= frase;  read(frase)      ;
          end;
       if p = pri    then borrar ( pri ,p)
                      else borrar (ult,p);
        long:=longitud – 1;
      end;
     ganador := pri^.item;
   end;
```

LISTAS CIRCULARES- Resolución con listas simples - borrar

```
procedure borrar ( var p, actual: juego);
var
    q : juego;
begin
    if p <> NIL then begin
                        q := actual;
                        actual:= actual^.sig;
                        p^.sig:=actual;
                        dispose( q )
                        end;
end;
```

JUEGO DE LA ORACION

- ✓ Solución con LISTAS CIRCULARES

Estructura de Datos LISTA CIRCULAR - VENTAJAS

- En las listas circulares, nunca se llega a una posición en la que ya no sea posible desplazarse (no existe el nil).
- Cuando se llegue al último elemento, el desplazamiento volverá a comenzar desde el primer elemento.

Estructura de Datos LISTA CIRCULAR - CARACTERISTICAS

Es una estructura de datos homogénea, dinámica y lineal.

Tienen un puntero a un elemento de la lista para poder iniciar su recorrido.

El puntero del último nodo apunta al primer nodo.

Tiene la misma estructura que una lista simple, pero varía en cuanto a la organización. Esto provocará que prestemos especial atención en sus operaciones

Estructura de Datos LISTA CIRCULAR - REPRESENTACION

```
TYPE
Cadena=string[50];
ListaC = ^ nodo;
Nodo = record
                dato:
     cadena;
                sig: listaC;
           end;
VAR  Pri: lista;
```

Igual estructura que las listas simples !!!

¿Cómo implementamos el juego?

LISTAS CIRCULARES- Juego

```
Program uno;
type
    niño  = record
          nombre:string;
          edad:integer;
    end;
    juego = ^nodo;
    nodo = record
                dato: niño;
                sig : juego;
           end,
```

```
Var
  p:juego;
  g:niño;

Procedure juegoOración(var j:juego;var  gan:niño);
…
Begin
  crearLista(p);
  juegoOración (p,g);
  write (g.nombre);
End;
```

LISTAS CIRCULARES- Juego

Puntos a considerar

- ¿Qué significa hacer una vuelta al juego?
- ¿Cuándo se termina?
- ¿Cómo elimino un jugador?

procedure juegoOración(var pri : juego; var ganador : niño);
Var ult, p : juego; frase, fraseAnterior: string;

```
begin
    p := pri; ult:= pri;
    while ( p^.sig <> p ) do begin
          fraseAnterior:=´ ´; readln (frase);
           while (EsCorrecta (frase, fraseAnterior) = true) do
             begin   ult := p;
                      p := p^.sig;
                     fraseAnterior:= frase;  readln (frase);
              end;
          borrar (ult,p);
    end;
    ganador:= p^.item;
    pri:= p;
end;
```

LISTAS CIRCULARES

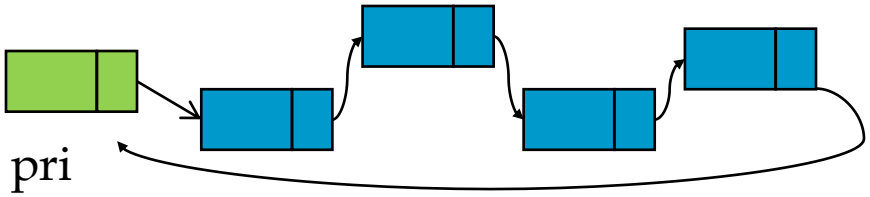
VENTAJAS

Cada nodo es accesible desde cualquier otro.

Simpleza para representar algunas situaciones de la vida real

DESVENTAJAS

En el recorrido se pueden producir lazos infinitos. Se puede encontrar una alternativa para identificar al primer nodo.



Estructura de Datos LISTA CIRCULAR - CONCLUSIONES

Las listas circulares nos simplifican la representación de algunas situaciones de la vida real, tal como vimos en la motivación.

Su característica principal es que nunca se llega a una posición en la que ya no sea posible desplazarse.

Sin embargo, se debe prestar especial atención en los recorridos, ya que se pueden producir lazos infinitos.

Estructura de Datos LISTA CIRCULAR – PARA PENSAR

Supongamos ahora que tenemos una lista circular con los procesos a ser atendidos por una CPU. La LISTA CIRCULAR está implementada con una lista circular pura donde el último elemento apunta físicamente al primero.

Cada proceso tiene un tiempo para finalizar su ejecución (lo llamaremos TP: tiempo del proceso).

Por otra parte, la CPU irá atendiendo de a un proceso por vez, durante un tiempo preestablecido (este tiempo de CPU lo llamamos: TCPU)

Si al atender el proceso se verifica que se terminó el tiempo TP, el proceso ha finalizado, de lo contrario queda en espera en la lista circular hasta poder ser ejecutado de nuevo. Atender un proceso significa que se accede al nodo de la lista, se le decrementa su tiempo de ejecuciónón de la siguiente manera: TP := TP –TCPU, si este tiempo es menor que cero se elimina ese nodo de la lista circular, en otro caso se lo deja en la lista esperando a volver a ser atendido, y es avanza al siguiente nodo.

Estructura de Datos LISTA CIRCULAR – PARA PENSAR

- ¿Qué datos contendría la lista circular en cada nodo?
-¿Cómo implementamos la situación planteada?

TRAER LA SOLUCION DEL ENUNCIADO DE LA PANTALLA ANTERIOR PARA LA PROXIMA CLASE