

FÓRMULAS TIEMPO DE EJECUCIÓN

El tiempo de ejecución del **IF**



Tiempo de evaluar la condición + tiempo del cuerpo.

Si hay else, Tiempo de evaluar la condición + max(then,else).

El tiempo de ejecución del **FOR**



$(3N + 2) + N(\text{cuerpo del for})$.

N representa la cantidad de veces que se ejecuta el for.

El tiempo de ejecución del **WHILE**



$C(N+1) + N(\text{cuerpo del while})$.

N representa la cantidad de veces que se ejecuta el while. ($N \geq 0$)

C cantidad de tiempo en evaluar la condición

El tiempo de ejecución del **REPEAT UNTIL**



$C(N) + N(\text{cuerpo del repeat})$.

N representa la cantidad de veces que se ejecuta el repeat. ($N > 0$)

C cantidad de tiempo en evaluar la condición

Clase 11-2

TIPOS DE DATOS

CADP – TIPOS DE DATOS - LISTA

SIMPLE: aquellos que toman un único valor, en un momento determinado, de todos los permitidos para ese tipo.

TIPO DE DATO

COMPUESTO: pueden tomar varios valores a la vez que guardan alguna relación lógica entre ellos, bajo un único nombre.

SIMPLE

COMPUESTO

DEFINIDO POR EL LENGUAJE

DEFINIDO POR EL PROGRAMADOR

DEFINIDO POR EL LENGUAJE

DEFINIDO POR EL PROGRAMADOR

Integer
Real
Char
Boolean
Puntero

Subrango

String

Registros

Arreglos

Lista

Clasi

LISTAS

- AGREGAR ADELANTE

```
procedure agregarAdelante (var pI:listaE; num:integer);
Var
  nuevo:listaE;  Creo espacio para el
                  nuevo elemento
Begin
  new (nuevo); nuevo^.elem:= num; nuevo^.sig:=nil;

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

Evalúo el caso y reasigno los punteros

- AGREGAR ATRÁS

```
procedure agregarAlFinal2 (var pI,pU:listaE; num:integer);
Var
  nuevo:listaE;

Begin
  new (nuevo); nuevo^.elem:= num; nuevo^.sig:=nil;

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

Evalúo si la lista está vacía

Actualizo el siguiente del último nodo y al último nodo

- BÚSQUEDA

- DESORDENADA

```

function buscar (pI: listaE; valor:integer):boolean;
Var
  aux:listaE;
  encontré:boolean;

Begin
  encontré:= false;
  aux:= pI;
  while ((aux <> nil) and (encontré = false)) do
    begin
      if (aux^.elem = valor) then
        encontré:=true
      else
        aux:= aux^.sig;
      end;
    buscar:= encontré;
  end;

```

Qué modificación

○ ORDENADA

```

function buscar (pI: listaE; valor:integer):boolean;
Var
  aux:listaE;
  encontré:boolean;

```

Funciona si el
recibo es

```

Begin
  encontré:= false;
  aux:= pI;
  while ((aux <> nil) and (aux^.elem < valor)) do
    begin
      aux:= aux^.sig;
    end;

```

Es necesario r
orden de las co
Neces

```

  if (aux <> nil) and (aux^.elem = valor) then encontré:= true;

  buscar:= encontré;
end;

```

Buscar en una lista tiene las mismas

- **ELIMINAR**
 - **DESORDENADA**

```

procedure eliminar (Var pI: listaE; valor:integer);
Var
  actual,ant:listaE;

Begin
  actual:=pI;
  while (actual <> nil) and (actual^.elem <> valor) do begin
    ant:=actual;
    actual:= actual^.sig;
  end;
  if (actual <> nil) then
    if (actual = pI) then
      pI:= pI^.sig;
    else
      ant^.sig:= actual^.sig;

      dispose (actual);
End;

```

Qué modifico si el
puede repet

CON ELEMENTOS QUE SE REPITEN

```

procedure eliminar (Var pI: listaE; valor:integer);
Var
  actual,ant:listaE;

Begin
  actual:=pI;
  while (actual <> nil) do begin
    if (actual^.elem <> valor) then begin
      ant:=actual;  actual:= actual^.sig;
    end;
    else begin
      if (actual = pI) then
        pI:= pI^.sig;
      else
        ant^.sig:= actual^.sig;
        dispose (actual);
        actual:= ant;
      end;
End;

```

Qué mo
ordenad