

Introducción a la programación

Explicación P0

CADP 2023



FACULTAD DE INFORMATICA



UNIVERSIDAD
NACIONAL
DE LA PLATA

Hasta ahora

En el curso de ingreso, trabajamos con el entorno R-info y su sintaxis acotada para crear programas.



¿Cómo se estructura un programa en Pascal?

R-Info

```
programa ejemploRInfo
areas
    ciudad: areaC(1,1,100,100)
robots
    robot robot1
variables
    {variables del programa}
comenzar
    {cuerpo del programa}
fin
variables
    Rinfo: robot1
comenzar
    AsignarArea(Rinfo,ciudad)
    iniciar(Rinfo,1,1)
fin
```

Pascal

```
program ejemploPascal;
var
    {variables del programa}
begin
    {cuerpo del programa}
end.
```

Más adelante, veremos
ejemplos

¿Qué tipos de variables existen en Pascal?

R-Info
Pascal

Tipo de variable	Datos
Numero Boolean	Números enteros V - F
Integer Real Boolean <i>Otros ...</i>	Números enteros Números reales True - False

¿Cómo se declaran variables en Pascal?

R-Info

```
programa ejemploRInfo
areas
    ciudad: areaC(1,1,100,100)
robots
    robot robot1
variables
    nombre_variable: tipo
comenzar
    {cuerpo del programa}
fin
variables
    Rinfo: robot1
comenzar
    AsignarArea(Rinfo,ciudad)
    iniciar(Rinfo,1,1)
fin
```

Pascal

```
program ejemploPascal;
var
    nombre_variable: tipo
begin
    {cuerpo del programa}
end.
```

Más adelante, veremos
ejemplos

¿Cómo se da valor a una variable?

R-info { Usando el **operador** :=

Pascal { Usando el **operador** :=
Mediante la **operación de lectura** de teclado: `read(variable)`

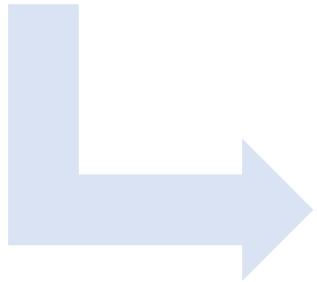
¿Cómo se imprime el valor de una variable?

R-info { Usando la instrucción: `Informar(variable)`

Pascal { Mediante la **operación de escritura** de pantalla: `write(variable)`

Veamos un ejemplo en Pascal

Implementar un programa en *Pascal* que lea de teclado dos números enteros, realice la suma de los mismos e imprima en pantalla el resultado obtenido.



```
program ejercicio;

var
    num1, num2, suma: integer;

begin
    read(num1);
    read(num2);
    suma := num1 + num2;
    write('El resultado es: ', suma);
end.
```

PARA PENSAR

- ¿Qué modificaciones deberían hacerse en el programa si se quisiera informar, además del resultado obtenido, los números que fueron sumados?
- ¿Y si se quisiera informar el doble del resultado obtenido?



Estructuras de control

Explicación P1

CADP 2023



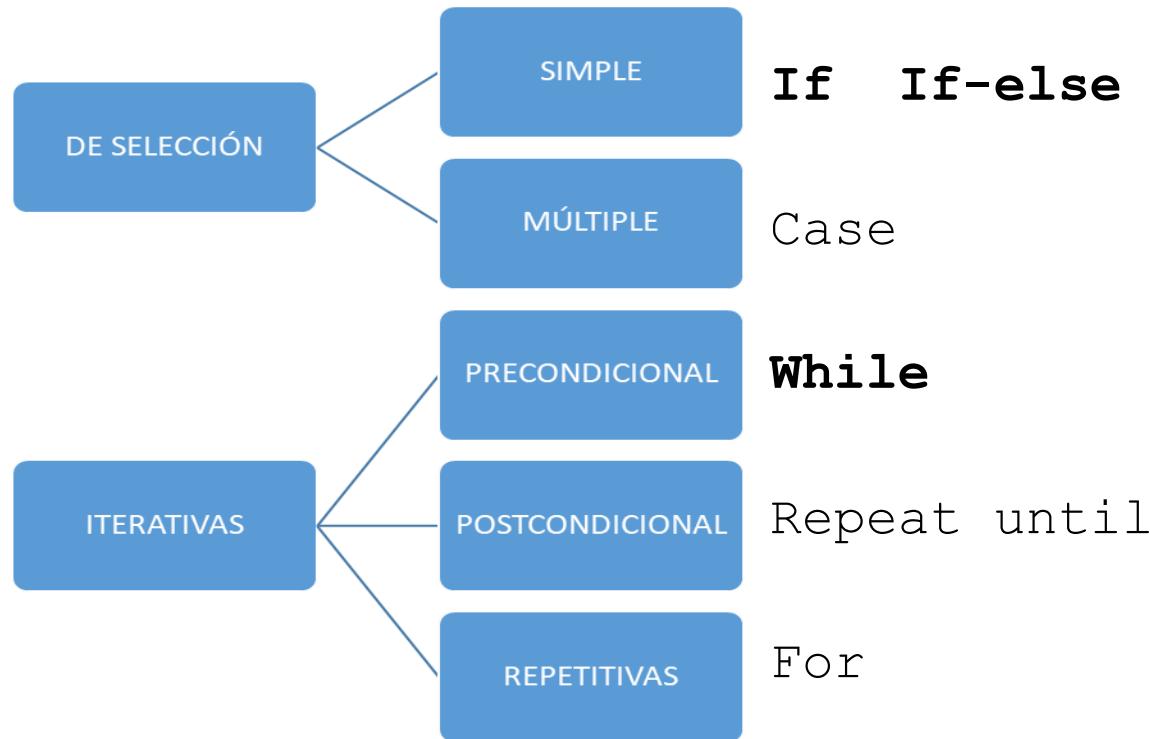
FACULTAD DE INFORMATICA



UNIVERSIDAD
NACIONAL
DE LA PLATA

ESTRUCTURAS DE CONTROL - *Pascal*

Lo visto en teoría



En esta práctica trabajaremos con *if* y *while*



EJEMPLOS DE USO

Estructura de control: if

Realice un programa que lea de teclado dos números enteros e informe el resultado de la suma de ambos, **sólo si éste es mayor que 50.**

```
program sumaMayor50;
var
    numero1, numero2, res: integer;
begin
    readln(numero1);
    readln(numero2);
    res := numero1 + numero2;
    if (res > 50)then
        writeln('El resultado es: ', res);
end.
```

EJEMPLOS DE USO

Estructura de control: if - else

Realice un programa que lea de teclado un número entero que representa la nota de un examen final e informe **si el alumno aprobó o no**. Considere que este examen se aprueba con 4 o más.

```
program notaExamen;
var
    nota: integer;
begin
    readln(nota);
    if (nota >= 4) then
        writeln('El alumno aprobó')
    else
        writeln('El alumno no aprobó');
end.
```

EJEMPLOS DE USO

Estructura de control: while

Realice un programa que lea de teclado números enteros **hasta que se ingrese el 0 (cero)** e informe la cantidad de números leídos.

```
Program numeros;
Var
    numero, cant: integer;
Begin
    cant:= 0;
    read(numero);
    while (numero <> 0) do begin
        cant:= cant +1;
        read(numero);
    end;
    writeln('La cantidad de números leídos es: ',cant);
End.
```

EJERCICIO

Realizar un programa que lea de teclado números enteros **hasta que se ingrese el 0 (cero)** e informe la cantidad de **números mayores que 5**.

```
Program ejercicioExp1;
var
    numero, cant: integer;
begin
    cant := 0;
    read(numero);
    while(numero <> 0) do begin
        if(numero > 5) then
            cant:= cant + 1;
        read(numero);
    end;
    write('La cantidad de números mayores que 5 es', cant);
end.
```

FIN

Estructuras de control

Explicación P1 (parte 2)

CADP 2023



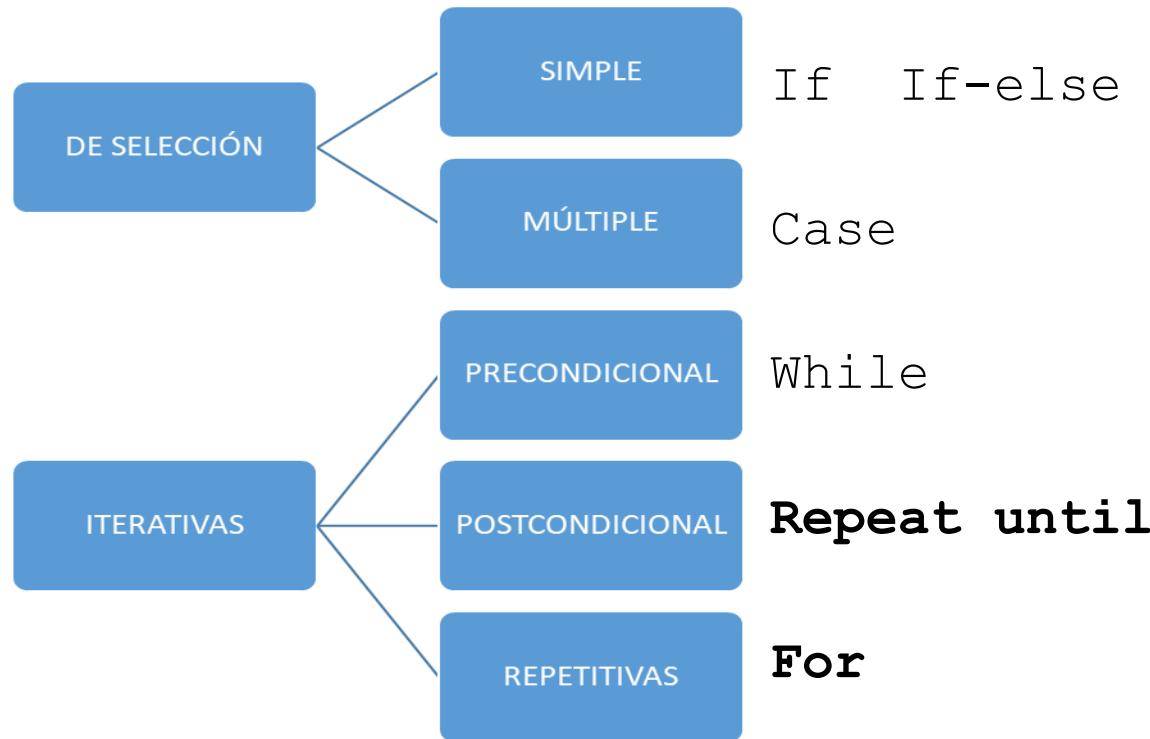
FACULTAD DE INFORMATICA



UNIVERSIDAD
NACIONAL
DE LA PLATA

ESTRUCTURAS DE CONTROL - *Pascal*

Lo visto en teoría



En esta práctica trabajaremos con *Repeat until* y *for* ➔

EJEMPLOS DE USO

Estructura de control: for

Realice un programa que lea de teclado 10 números enteros e informe el resultado de la suma.

```
Program suma;
Var
    i, numero, res: integer;
Begin
    res := 0;
    for i:= 1 to 10 do begin
        readln(numero);
        res:= res + numero;
    end;
    writeln('La suma es:', res);
End.
```

PARA RESOLVER

Estructura de control: for

¿Qué imprime el siguiente código?

```
program queImprime;
var
    i: integer;
begin
    for i:= 1 to 5 do
        writeln(i);
end.
```

El índice de un for no debe modificarse. ¿Qué pasa si ejecutamos el siguiente código?



```
program infinito;
var
    i: integer;
begin
    for i:= 1 to 5 do begin
        writeln(i);
        i:= 1;
    end;
    readln();
end.
```

PARA RESOLVER

Estructura de control: for

¿Qué imprime el siguiente código?

```
Program queImprime2;
Var
    i: integer;
Begin
    for i:= 1 to 5 do
        if ((i mod 2) = 0) then
            writeln(i);
End.
```

PARA RESOLVER

Estructura de control: Repeat until

Realice un programa que **lea** una secuencia de números hasta leer un número mayor a 100 **el cual debe procesarse**. Al finalizar se debe informar la suma de todos los números leídos.

```
Program ejemploRepeatuntil;
Var
    n, suma: integer;
Begin
    suma := 0;
    repeat
        readln(n);
        suma := suma + n;
    until (n > 100);
    writeln('La suma es:', suma);
End.
```

PARA RESOLVER

Estructura de control: Repeat until

Modifique el programa anterior para que lea una secuencia de números hasta leer el número 100, **el cual debe procesarse**. Al finalizar se debe informar la suma de todos los números leídos.

```
Program ejemploRepeatuntil;
Var
    n, suma: integer;
Begin
    suma := 0;
    repeat
        readln(n);
        suma := suma + n;
    until ( n = 100 );
    writeln('La suma es:', suma);
End.
```

Estructuras de control

Cálculo de máximos

Explicación P1 (parte 3)

CADP 2021



FACULTAD DE INFORMATICA



UNIVERSIDAD
NACIONAL
DE LA PLATA

CALCULAR VALOR MÁXIMO

Realizar un programa que lea números naturales desde teclado. La lectura debe finalizar cuando se ingrese el número 0, el cual **no** debe procesarse.

Informar en pantalla cuál es el número máximo leído.

¿Estructura de control?

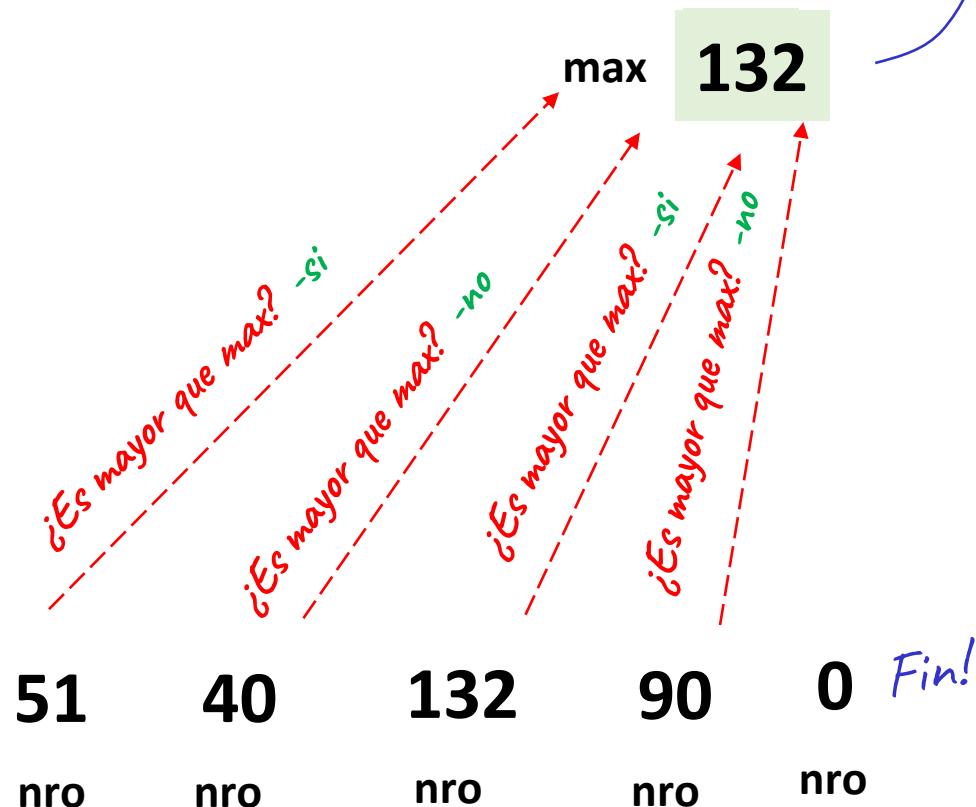
¿Datos a calcular?

¿Datos a leer de teclado ?

CALCULAR VALOR MÁXIMO

Analizando el problema...

Variable para llevar el máximo,
inicializada con un valor muy
bajo.



Se comienza a leer
hasta que llegue el 0...

El número leido más alto fue: 132

CALCULAR VALOR MÁXIMO

Realizar un programa que lea números naturales desde teclado. La lectura debe finalizar cuando se ingrese el número 0, el cual **no** debe procesarse.

Informar en pantalla cuál es el número máximo leído.

```
program valorMaximo;
var
    nro, max: integer;
Begin
    max:= -1;
    readln(nro); {leo un número}
    while (nro <> 0) do begin
        if (nro > max) then {evalua el máximo}
            max:= nro;
        readln(nro); {leo otro número}
    end;
    writeln('El número más alto fue: ', max);
end.
```

CALCULAR VALOR MÁXIMO

Realizar un programa que lea números naturales desde teclado. La lectura debe finalizar cuando se ingrese el número 0, el cual **no** debe procesarse.

Informar en pantalla cuáles son los 2 número máximos leídos.

¿Estructura de control?

¿Datos a calcular?

¿Datos a leer de teclado ?

CALCULAR 2 VALORES MÁXIMOS

Analizando el problema...

Variables para llevar el máximo1
y el máximo2.

max1 **132**
max2 **51**

Se comienza a leer
hasta que llegue el 0...



Los 2 números más alto fueron: 132 y 51

CALCULAR 2 MÁXIMOS

```
program DosMaximos;
var
  max1, max2: integer;
  n: integer;
begin
  max1:=-1; max2:=-1; {inicializa Los máximos}
  read(n);
  while (n <> 0) do begin
    if (n > max1) then begin {evalua máximo 1}
      max2:=max1;
      max1:=n;
    end
    else
      if (n > max2) then {evalua máximo 2}
        max2:=n;
    read(n);
  end;
  writeln('Los 2 números mas altos fueron', max1, 'y', max2);
end.
```

Estructuras de control

Cálculo de máximos - *Ejemplo*

Explicación P1 (parte 4)

CADP 2023



FACULTAD DE INFORMATICA



UNIVERSIDAD
NACIONAL
DE LA PLATA

CALCULAR 2 MÁXIMOS

Se leen las alturas de **20** jugadores de básquet junto con su DNI.
Informar los DNI de los 2 jugadores más altos.

¿Estructura de control?

¿Datos a calcular?

¿Datos a leer de teclado ?

CALCULAR 2 MÁXIMOS

Entendiendo el problema...



		Jug 1	Jug 2	Jug 3	Jug 4	Jug 5
DNI		31.111.333	30.222.888	35.666.111	26.777.000	24.111.555
ALTURA		1,78	1,75	1,98	1,68	1,85
MAX1	-1	1,78	1,78	1,98	1,98	1,98
DNI MAX1	0	31.111.333	31.111.333	35.666.111	35.666.111	35.666.111
MAX2	-1	-1	1,75	1,78	1,78	1,85
DNI MAX2	0	0	30.222.888	31.111.333	31.111.333	24.111.555

Evaluación y modificación
de 2 máximos.

```
if(altura > max1)then
begin
  max2:= max1;
  dnimax2:= dnimax1;
  max1:= altura;
  dnimax1:= dni;
end
else
  if(altura > max2)then
  begin
    max2:= altura;
    dnimax2:= dni;
  end;
```

CALCULAR 2 MÁXIMOS

```
program basquet;
var
    altura, max1, max2: real;
    dni, dnimax1, dnimax2: integer;
    i: integer;
begin
    max1:=-1; max2:=-1; {inicializo Los máximos}
    for i:=1 to 20 do begin
        read(altura);
        read(dni);
        if (altura > max1) then begin {evalúo máximo 1}
            max2:=max1;
            dnimax2:=dnimax1;
            max1:=altura;
            dnimax1:=dni;
        end
        else
            if (altura > max2) then begin {evalúa máximo 2}
                max2:=altura;
                dnimax2:=dni;
            end;
    end;
    writeln('DNI 1er jugador más alto: ', dnimax1);
    writeln('DNI 2do jugador más alto: ', dnimax2);
end.
```

Modularización

Explicación P2

CADP 2023



FACULTAD DE INFORMATICA

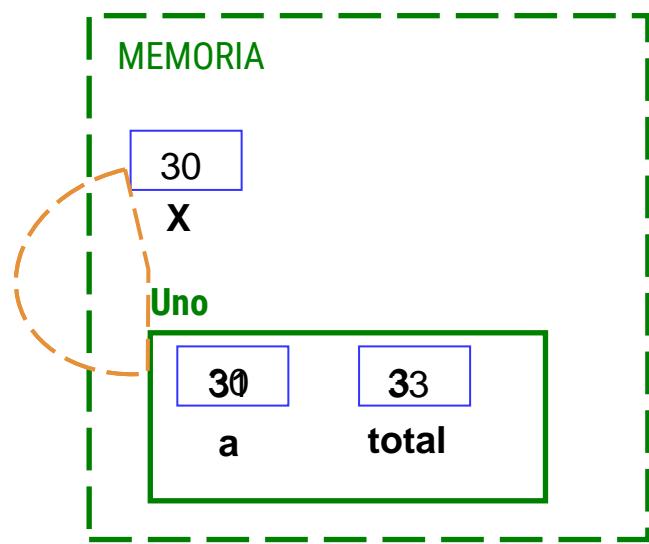


UNIVERSIDAD
NACIONAL
DE LA PLATA

MODULARIZACIÓN

Analicemos la ejecución del siguiente código

```
Program paramValor;
→ Procedure uno (a: integer);
  var
    → total: integer;
  Begin
    → total:= 3;
    → total:= total + a;
    → a:= a + 1;
    → writeln ('El valor de total es: ',total);
    → writeln ('El valor de a es: ', a);
  end;
  var
    → x: integer;
  begin
    → x:= 30;
    → uno(x);
    → writeln ('El valor de x es: ', x);
    → readln;
  end.
```



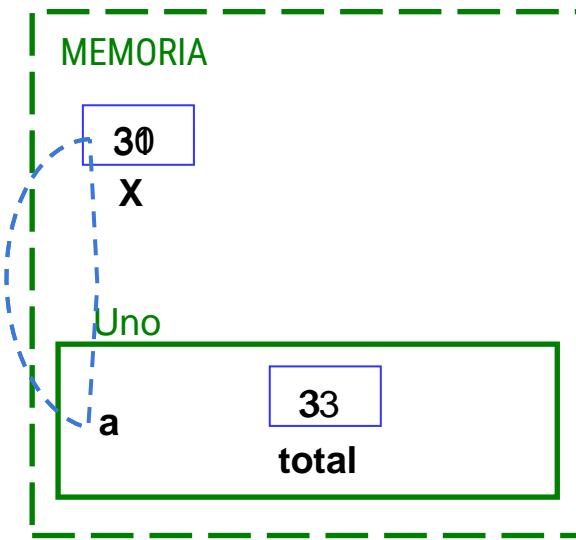
PANTALLA

```
El valor de total es: 33
El valor de a es: 31
El valor de x es: 30
```

MODULARIZACIÓN

Analicemos la ejecución del siguiente código

```
Program paramReferencia;
  Procedure uno (var a: integer);
    var
      total: integer;
    Begin
      total:= 3;
      total:= total + a;
      a:= a + 1;
      writeln ('El valor de total es: ',total);
      writeln ('El valor de a es: ', a);
    end;
    var
      x: integer;
    begin
      x:= 30;
      uno(x);
      writeln ('El valor de x es: ', x);
      readln;
    end.
```



PANTALLA

```
El valor de total es: 33
El valor de a es: 31
El valor de x es: 31
```

MODULARIZACIÓN

Ejercicio

- a) Realice un **procedimiento** que **reciba** como parámetro **un número** entero y **retorne** la cantidad de dígitos impares y la cantidad de dígitos pares que posee el número recibido.

- b) Utilizando el procedimiento definido en a) realice un programa que lea 20 números enteros e informe la cantidad de números que tienen más dígitos pares que impares.

¿Qué datos debemos comunicar entre el módulo y su llamador?

```
Procedure descomponer (num: integer; var cantP, cantI: integer);
```

MODULARIZACIÓN

VARIABLE LOCAL: Accesible sólo por el proceso

```
Procedure descomponer (num: integer; var cantP,cantI: integer);  
Var  
    dig: integer;  
Begin  
    cantP:= 0;  
    cantI:= 0;  
    while (num <> 0) do begin  
        dig:= num mod 10;  
        if ((dig mod 2) = 0) then  
            cantP:= cantP + 1  
        else  
            cantI:= cantI + 1;  
        num:= num div 10;  
    end;  
end;
```

Solución

VARIABLES DEL PROG. PPAL

```
Program ejercicio;
Procedure descomponer (num: integer; var cantP,cantI: integer);
{... aquí va el cuerpo del procedure descomponer ... }

var
  i, num, pares, impares: integer;
  cant: integer;
begin
  cant:= 0;
  for i:= 1 to 20 do begin
    readln(num);
    descomponer(num, pares, impares);
    if (pares > impares) then
      cant:= cant + 1;
  end;
  writeln('La cant. de nros que tienen mas dig pares que imp es:', cant);
end.
```

Modularización

Procesamiento de secuencias de caracteres

Explicación P2 (parte 2)

CADP 2023



FACULTAD DE INFORMATICA



UNIVERSIDAD
NACIONAL
DE LA PLATA

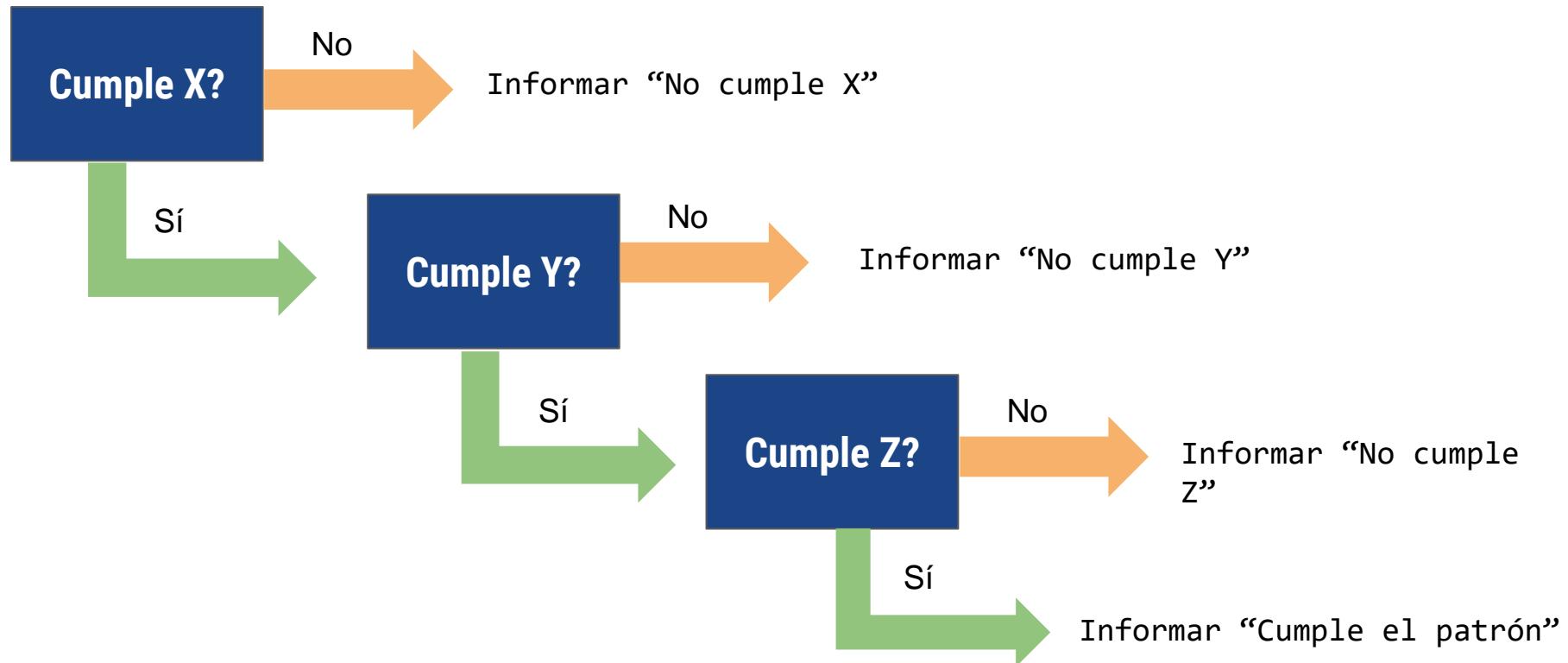
Ejercicio

Realizar un programa que lea una secuencia de caracteres y verifique si cumple con el patrón **X&Y&Z*** donde:

- **X** es una secuencia de caracteres numéricos
- **Y** es una secuencia de vocales minúsculas
- **Z** es una secuencia de caracteres del doble de longitud de **Y**
- Los caracteres & y * seguro existen

Nota: en caso de no cumplir, informar que parte del patrón no se cumplió

Pasos para la solución



Implementación

Chequear si cumple X

```
function esNumero(c: char): boolean;
begin
    esNumero := (c >= '0') and (c <= '9');
end;

procedure cumpleX(var cumple : boolean);
var
    c:char;
begin
    writeln('Ingrese la secuencia X');
    readln(c);
    while (c <> '&') and (cumple) do begin
        if (not esNumero(c)) then
            cumple := false
        else
            readln(c);
    end;
end;
```

Implementación

Chequear si cumple Y

```
function esVocalMinuscula(c: char): boolean;
begin
    esVocalMinuscula:=(c='a')or(c='e')or(c='i')or(c='o')or(c='u');
end;

procedure cumpleY(var cumple:boolean; var long:integer);
var
    c : char;
begin
    writeln('Ingrese la secuencia Y');
    readln(c);
    while (c <> '&') and (cumple) do begin
        if (not esVocalMinuscula(c)) then
            cumple := false
        else begin
            long := long + 1;
            readln(c);
        end;
    end;
end;
```

Implementación

Chequear si cumple Z

```
procedure cumpleZ(long:integer; var cumple:boolean);
var
  c : char;
  longZ : integer;
begin
  longZ := 0;
  writeln('Ingrese la secuencia Z');
  readln(c);
  while (c <> '*')do begin
    longZ := longZ + 1;
    readln(c);
  end;
  cumple := (longZ = long*2);
end;
```

Implementación

Programa principal

```
Program XYZ;
  {...Procedures y functions definidos previamente ...}
var
  long : integer;
  cumple : boolean;
begin
  cumple := true;
  cumpleX(cumple);
  if (cumple) then begin      { if x }
    long := 0;
    cumpleY(cumple, long);
    if (cumple) then begin    { if Y }
      cumpleZ(long,cumple);
      if (cumple) then       { if z }
        writeln('Se cumple la secuencia')
      else
        writeln('No cumple con Z')
    end                      { end del if y }
    else
      writeln('No cumple con Y');
  end                      { end del if x }
  else
    writeln('No cumple con X');
end.
```

Registros

Explicación P3

CADP 2023



FACULTAD DE INFORMATICA



UNIVERSIDAD
NACIONAL
DE LA PLATA

REGISTROS

Definición de tipo

```
tipoRegistro = record  
    campo1: tipo-campo1;  
    campo2: tipo-campo2;  
end;
```

Declaración de variable

```
miVariable: tipoRegistro;
```

Acceso a los datos

```
miVariable.campo1
```

Operaciones

- **Leer un registro:** campo a campo
- **Imprimir un registro:** campo a campo
- **Comparar un registro con otro:** campo a campo
- **Asignar un registro a otro:** usando el operador :=

Ej: $r1 := r2;$ ($r1$ y $r2$ son del mismo tipo de registro)

REGISTROS

Ejercicio

Se lee desde teclado una secuencia de sitios turísticos hasta que llegue el sitio con nombre ‘fin’. De cada sitio se conoce: nombre, provincia a la que pertenece, cantidad de actividades, cantidad de personas que lo visitaron en un mes.

Se pide:

- a) El nombre del sitio con mayor cantidad de actividades.
- b) La cantidad de sitios con más de 20.000 personas que lo visitaron en un mes.

REGISTROS

A continuación se muestra una lectura de sitios turísticos. Notar cómo se actualizan los valores correspondientes de acuerdo a los datos leídos y analizar por qué en algunas situaciones los valores no se actualizan.

Nombre: Necochea
Provincia: Bs. As.
Cant. Act.: 10
Cant. Visitas: 15.000

Nombre: Ushuaia
Provincia: Tierra del Fuego
Cant. Act.: 9
Cant. Visitas: 21.000

Nombre: Bariloche
Provincia: Rio Negro
Cant. Act.: 20
Cant. Visitas: 20.000

Nombre: San Rafael
Provincia: Mendoza
Cant. Act.: 15
Cant. Visitas: 10.000

Nombre: fin
Provincia:
Cant. Act.:
Cant. Visitas:

Sitio turístico con mayor cantidad de actividades:

Bariloche
20

Cantidad de sitios con más de 20000 visitas:

1

REGISTROS

Implementación

```
Program secretaria;

Type
    sitio = record
        nombre: string;
        prov: string;
        cantAct: integer;
        cantVis: integer;
    end;

Var
    sitioTur: sitio;
    cant, max: integer;
    nomMax: string;
```

REGISTROS

Implementación

```
{Programa Principal}  
Begin  
    cant:= 0;  
    nomMax:= " ";  
    max:= -1;  
    LeerRegistro(sitioTur); {Se lee el primer registro}  
    While (sitioTur.nombre <> 'fin') Do begin  
        actualizarMax(sitioTur.cantAct, sitioTur.nombre, max, nomMax);  
        if (sitioTur.cantVis > 20000) then  
            cant:= cant + 1;  
        LeerRegistro(sitioTur); {se lee otro registro}  
    end;  
    Write ("Sitio con mas actividades: ", nomMax);  
    Write ("Sitios con más de 20000 visitas: ", cant);  
End.
```

REGISTROS

Implementación

```
Procedure LeerRegistro (var s:sitio);
begin
  With s do begin
    readln(nombre);
    if (nombre <> 'fin') then
      begin
        readln(prov);
        readln(cantAct);
        readln(cantVis);
      end;
    end;
  end;
```

REGISTROS

Implementación

```
{Programa Principal}  
Begin  
    cant:= 0;  
    nomMax:= " ";  
    max:= -1;  
    LeerRegistro(sitioTur); {Se lee el primer registro}  
    While (sitioTur.nombre <> 'fin') do begin  
        actualizarMax(sitioTur.cantAct, sitioTur.nombre, max, nomMax);  
        if (sitioTur.cantVis > 20000) then  
            cant:= cant + 1;  
        LeerRegistro(sitioTur); {se lee otro registro}  
    end;  
    Write ("Sitio con mas actividades: ", nomMax);  
    Write ("Sitios con más de 20000 visitas: ", cant);  
End.
```

REGISTROS

Implementación

```
Procedure actualizarMax (cantAct: integer; nombreAct: string;
                           var max: integer; var nommax: string );
begin
  if (cantAct > max) then
    begin
      max:= cantAct;
      nommax:= nombreAct;
    end;
  end;
```

Registros

Corte de control

Explicación P3 (parte 2)

CADP 2023



FACULTAD DE INFORMATICA



UNIVERSIDAD
NACIONAL
DE LA PLATA

CORTE DE CONTROL

Ejercicio

Realice un programa que lea de teclado la información de ciudades turísticas (provincia, nombre de ciudad, cantidad de actividades, cantidad de visitantes), ORDENADA POR PROVINCIA, que termina con la provincia 'fin'.

Obtener cuál es la provincia con más visitantes a sus ciudades turísticas.

CORTE DE CONTROL - *Analizando el problema*

Se lee la primera ciudad

Provincia: Bs. As.

Nombre: Necochea

Cant. Act.: 10

Cant. Visitas: 15.000

Provincia: Bs. As

Nombre: La Plata

Cant. Act.: 9

Cant. Visitas: 21.000

La ciudad leída NO pertenece
a la prov en proceso

Provincia: Santa Cruz

Nombre: El Chaltén

Cant. Act.: 20

Cant. Visitas: 20.000

Provincia: Santa Cruz.

Nombre: El Calafate

Cant. Act.: 15

Cant. Visitas: 35.000

Provincia: fin

Nombre:

Cant. Act.:

..... , . ,

Se almacena Bs. As. como la
prov que está en proceso.

Se acumula la cant de vi
para la prov en proceso

Como se terminó de procesar la prov
actual, se actualiza el máximo

Prov actual

Santa Cruz

Cant. total de
visitantes de la prov
actual

350000

Prov con mayor
cant de visitantes

BsAs Cruz

cant de visitantes

3550000

CORTE DE CONTROL - *Implementación*

```
Program secretaria;

Type
  tciudad = record
    prov: string;
    nombre: string;
    cantAct: integer;
    cantVis: integer;
  end;

var
  ciudadTur: tciudad;
  max, cantidad: integer;
  nomMax, provActual: string;
```

CORTE DE CONTROL - *Implementación*

```
Program secretaria;
```

Type

```
  tciudad = record
    prov: string;
    nombre: string;
    cantAct: integer;
    cantVis: integer;
  end;
```

var

```
  ciudadTu
  max, cant
  nomMax, provActual: string;
```

¿ Por qué NO se debe
leer nuevamente
dentro de este while ?

```
begin {Programa Principal}
  nomMax:= '';
  max:= -1;
  leer(ciudadTur); {Se Lee 1er registro}
  while (ciudadTur.prov <> 'fin') do begin
    provActual:= ciudadTur.prov;
    cantidad:=0;
    while (ciudadTur.prov = provActual) do begin
      cantidad:= cantidad + ciudadTur.cantVis;
      leer(ciudadTur); {Se Lee otro registro}
    end;
    actualizarMax(cantidad, provActual, max, nomMax);
  end;
  writeln('Prov con m');
end.
```

¿ Por qué NO es correcto
enviar ciudadTur.Prov ?

CORTE DE CONTROL - *Implementación*

```
procedure leer(var c: tciudad);
begin
  With c do begin
    readln(prov);
    if (prov <> 'fin') then begin
      readln(nombre);
      readln(cantAct);
      readln(cantVis);
    end;
  end;
end;
```

```
Procedure actualizarMax(cantAct: integer; nombreAct: string; var max:
                           integer; var nommax: string);
Begin
  if (cantAct > max) then begin
    max:= cantAct;
    nommax:= nombreAct;
  end;
End;
```

VECTORES

EXPLICACIÓN PRÁCTICA 4

CADP 2023

Aspectos básicos

Ejemplo:

Type

```
vector = Array [1..10] of  
integer;  
Var  
  v: vector;  
  d1: integer;
```

V	10	14	19	25	33					
	1	2	3	4	5	6	7	8	9	10

Dimensión física = 10
Dimensión lógica = 5

¿Cómo accedo al elemento
de la posición 5?

¿Qué operaciones puedo
hacer con **v[5]**?

VECTORES DE NÚMEROS

Ejercicio 1

Realizar un programa que cargue un vector de 1500 números enteros positivos.
Al finalizar la carga informe la posición de los números mayores que 50.

¿Dónde almaceno los números?

10	3	50	88	21	93	33	52	...	90
1	2	3	4	5	6	7	8	...	1500

¿Necesito llevar la dimensión lógica?

¿Qué datos debo informar?

VECTORES DE NÚMEROS

Solución del ejercicio 1

```
Program Ejercicio1;
```

```
Type
```

```
    rango = 1..1500;
```

```
    numeros = array [rango] of integer;
```

{ ... Acá se declaran los módulos }

```
var
```

```
    v: numeros;
```

```
begin
```

```
    cargar(v);
```

```
    procesar(v);
```

```
end.
```

```
procedure cargar (var v: numeros);  
var  
    i:rango;  
begin  
    for i:= 1 to 1500 do  
        read(v[i]);  
end;
```

```
procedure procesar(v: numeros);  
var  
    i: rango;  
begin  
    for i:= 1 to 1500 do begin  
        if (v[i] > 50) then  
            writeln('el nro en la posición',i, 'es > 50');  
    end;  
end;
```

VECTORES DE NÚMEROS

Ejercicio 1b

Modifique el ejercicio 1 para terminar la carga de números cuando se lee el número 0 que no debe procesarse, o se complete el vector de 1500.

¿Qué estructura de control necesito para realizar la carga?

¿Necesito manejar la dimensión lógica?

Solución del ejercicio 1b

VECTORES DE NÚMEROS

Program Ejercicio1;

Type

rango = 1..1500;

numeros = array [rango] of
integer;

{ ... Acá se declaran los módulos
... }

var

v: numeros;

dimLog: integer;

begin

cargar(v, dimLog);

procesar(v, dimLog);

end.

```
procedure cargar (var v: números;  
                  var dl: integer);  
  
var  
    n: integer;  
  
Begin  
    dl:= 0  
    read(n);  
    while (n <> 0) and (dl < 1500) do begin  
        dl:= dl +1;  
        v[dl] := n;  
        read(n);  
    end;  
end;
```

```
procedure procesar(v:números, dl: integer);  
  
var  
    i: rango;  
  
Begin  
    i:= 1;  
    while (i <= dl) do begin  
        if (v[i] > 50) then  
            writeln('el nro en la posición', i, 'es > 50');  
        i:= i+1;  
    end;  
end;
```

VECTORES DE REGISTROS

Ejercicio 2

Realizar un programa que cargue un vector de 100 productos. De cada producto se conoce código, descripción y precio. **Al finalizar la carga** informe la cantidad de productos con precio mayor que 50.

¿Dónde almaceno los productos?

¿Qué estructura de control necesito para realizar la carga?

¿Necesito manejar la dimensión lógica?

¿Qué datos debo informar?

Solución del ejercicio 2

```
Program Ejercicio2;
```

Type

```
rango = 1..100;
```

```
producto = record
```

```
    cod: integer;
```

```
    desc: string;
```

```
    precio: real;
```

```
end;
```

```
vecProductos = array [rango] of  
producto;
```

var

```
vp: vecProductos;
```

```
cant: integer;
```

begin

```
cargar(vp);
```

```
procesar(vp, cant);
```

```
writeln(cant);
```

end.

VECTORES DE NUMEROS

```
procedure cargar (var v: vecProductos);  
  
var  
    i:rango;  
  
begin  
    for i:= 1 to 100 do  
        leer(v[i]);  
end;
```

Módulo de lectura
del registro producto

```
procedure procesar(vp:vecProductos;  
                    var cant:  
                    integer);  
  
var  
    i: rango;  
  
Begin  
    cant:= 0;  
    for i:= 1 to 100 do begin  
        if (vp[i].precio > 50) then  
            cant:= cant +1;  
    end;  
end;
```

VECTORES DE REGISTROS

Para analizar y resolver en clase

Ejercicio 2b

Modifique el ejercicio 2 para que la carga de productos finalice cuando se lea el producto con código 00 (que no debe procesarse) o se complete el vector de 100 productos.

¿Dónde almaceno los productos?

¿Qué estructura de control necesito para realizar la carga?

¿Necesito manejar la dimensión lógica?

¿Qué datos debo informar?

VECTORES DE REGISTROS

EXPLICACIÓN PRÁCTICA 4

CADP 2023

VECTORES DE REGISTROS

Ejercicio 1

Realizar un programa que cargue un vector de 100 productos. De cada producto se conoce código, descripción y precio. **Al finalizar la carga** informe la cantidad de productos con precio mayor que 50.

¿Dónde almaceno los productos?

¿Qué estructura de control necesito para realizar la carga?

¿Necesito manejar la dimensión lógica?

¿Qué datos debo informar?

Solución del ejercicio 1

VECTORES DE REGISTROS

Program Ejercicio2;

Type

rango = 1..100;

producto = record

 cod: integer;

 desc: string;

 precio: real;

end;

vecProductos = array [rango] of producto;

var

 vp: vecProductos;

 cant: integer;

begin

cargar(vp);

procesar(vp, cant);

writeln(cant);

end.

```
procedure cargar (var vp: vecProductos);  
var  
  i:rango;  
begin  
  for i:= 1 to 100 do  
    leer(vp[i]);  
end;
```

Módulo de lectura
del registro producto

```
procedure procesar(vp:vecProductos; var cant:integer);  
var  
  i: rango;  
Begin  
  cant:= 0;  
  for i:= 1 to 100 do begin  
    if (vp[i].precio > 50) then  
      cant:= cant +1;  
  end;  
end;
```

VECTORES DE REGISTROS

Para analizar y resolver en clase

Ejercicio 1b

Modifique el ejercicio 2 para que la carga de productos finalice cuando se lea el producto con código 00 (que no debe procesarse) o se complete el vector de 100 productos.

¿Dónde almaceno los productos?

¿Qué estructura de control necesito para realizar la carga?

¿Necesito manejar la dimensión lógica?

¿Qué datos debo informar?

VECTORES CONTADORES

EXPLICACIÓN PRÁCTICA 4 **continuación**

CADP 2023

Ejemplo

Se lee una secuencia de dígitos (números entre el 0 y el 9) hasta que se ingresa el cero, que debe procesarse. Al finalizar la secuencia, informar la cantidad de veces que aparece cada dígito.

Secuencia de prueba: 4 6 9 8 4 9 5 7 1 9 0

```
var
  num; cant0, cant1, cant2, cant3, ..., cant9 : integer;
begin
  cant0 := ...; cant1 := 0; cant2 := ...
repeat
  read(num);
  if (num=0) then
    cant0 := ...
  else if (num = 1) then cant1 := ...
  until (num = 0);
```

```
type
  vdigitos = array[0..9] of integer;
var
  digitos : vdigitos;
  num : integer;
begin
  inicializar(digitos);
  repeat
    read(num);
    digitos[num] := digitos[num] + 1;
  until (num = 0)
  imprimir(digitos);
end.
```

0	0	0	0	0	0	0	0	0	0
0	1	2	3	4	5	6	7	8	9

Ejercicio

Hacer un programa que lea una secuencia de números enteros terminada en 0. Informar la cantidad de veces que aparece cada dígito del 0 al 9 entre todos los números leídos.

Ejemplo: se leen los números: 457 9875 5 24879 0

{Se debe informar por cada dígito cuántas veces aparecieron cada uno de ellos}

0 aparece 0 veces;

1 aparece 0 veces;

2 aparece 1 veces;

3 aparece 0 veces;

4 aparece 2 veces;

5 aparece 3 veces;

6 aparece 0 veces;

7 aparece 3 veces;

8 aparece 2 veces;

9 aparece 2 veces;

¿Debo almacenar los números leídos?

¿Cómouento las veces que aparece cada dígito?

0	0	1	0	2	3	0	3	2	2
0	1	2	3	4	5	6	7	8	9

¿Necesito llevar la dimensión lógica?

```

Program Digitos;
Type
  rango=0..9;
  numeros=array [rango] of integer;

{Acá se declaran Los módulos}
var
  losnros:numeros;
  num:integer;
begin
  inicializar(losnros);
  read(num);
  while (num <> 0) do begin
    descomponer(losnros, num);
    read(num);
  end;
  informo(losnros);
end.

```

```

procedure inicializar(var a:numeros);
var i:rango;
begin
  for i:=0 to 9 do
    a[i]:=0;
end;

procedure descomponer(var a:numeros;
num:integer);
var
  resto:rango;
begin
  while (num <> 0) do begin
    resto:=num mod 10; {Obtengo digito}
    {Incremento contador asociado al digito}
    a[resto]:=a[resto] + 1;
    num:=num div 10; {Achico número}
  end;
end;
procedure informo(a:numeros);
var
  i:rango;
begin
  for i:=0 to 9 do
    writeln(i, ' = ',a[i]);
end;

```

1. Modifique para informar para cada número la cantidad de veces que aparece cada dígito.

Program Digitos;

Type

rango=0..9;
numeros=array [rango] of integer;

{Acá se declaran Los módulos}

var

losnros:numeros;
num:integer;

begin

inicializar(losnros);
 read(num);
 while (num <> 0) **do begin**
 descomponer(losnros, num);
 read(num);

end;

informo(losnros);

end.

```
procedure inicializar(var a:numeros);  
var i:rango;  
begin  
  for i:=0 to 9 do  
    a[i]:=0;  
end;  
  
procedure descomponer(var a:numeros;  
num:integer);  
var  
  resto:rango;  
begin  
  while (num <> 0) do begin  
    resto:=num mod 10; {Obtengo dígito}  
    {Incremento contador asociado al dígito}  
    a[resto]:=a[resto] + 1;  
    num:=num div 10; {Achico número}  
  end;  
end;  
procedure informo(a:numeros);  
var  
  i:rango;  
begin  
  for i:=0 to 9 do  
    writeln(i, ' = ', a[i]);  
end;
```

1. Modifique para informar para cada número la cantidad de veces que aparece cada dígito.

```
Program Digitos;
Type
  rango=0..9;
  numeros=array [rango] of integer;
{Acá se declaran Los módulos}
var
  losnros:numeros;
  num:integer;
begin
  inicializar(losnros);
  read(num);
  while (num <> 0) do begin
    descomponer(losnros, num);
    read(num);
  end;
  informo(losnros);
end.
```

```
procedure inicializar(var a:numeros);
var i:rango;
begin
  for i:=0 to 9 do
    a[i]:=0;
end;

procedure descomponer(var a:numeros;
num:integer);
var
  resto:rango;
begin
  while (num <> 0) do begin
    resto:=num mod 10; {Obtengo dígito}
    {Incremento contador asociado al dígito}
    a[resto]:=a[resto] + 1;
    num:=num div 10; {Achico número}
  end;
end;
procedure informo(a:numeros);
var
  i:rango;
begin
  for i:=0 to 9 do
    writeln(i, ' = ', a[i]);
end;
```

1. Modifique para informar para cada número la cantidad de veces que aparece cada dígito.

```
Program Digits;  
Type  
    rango=0..9;  
    numeros=array [rango] of integer;
```

{Acá se declaran los módulos}

```
var  
    losnros:numeros;  
    num:integer;  
begin  
    read(num);  
    while (num <> 0) do begin  
        inicializar(losnros);  
        descomponer(losnros, num);  
        informo(losnros);  
        read(num);  
    end;  
end.
```

```
Program Digitos;  
Type  
    rango=0..9;  
    numeros=array [rango] of integer;  
  
    {Acá se declaran Los módulos}  
var  
    losnros:numeros;  
    num:integer;  
begin  
    read(num);  
    while (num <> 0) do begin  
        inicializar(losnros);  
        descomponer(losnros, num);  
        informo(losnros);  
        read(num);  
    end;  
end.
```

2. Modifique para informar el dígito que más veces apareció para cada número

```

Program Digitos;
Type
  rango=0..9;
  numeros=array [rango] of integer;

{Acá se declaran Los módulos}
var
  losnros: numeros;
  num:integer;
begin
  read(num);
  while (num <> 0) do begin
    inicializar(losnros);
    descomponer(losnros, num);
    write('el dig que mas aparece es', DígitoMaximo(losnros));
    read(num);
  end;
end.

```

```

Function DígitoMaximo(a:numero):rango;
var
  i, digmax:rango;
  max:integer;
begin
  max:=-1;
  for i:=0 to 9 do
    if (a[i] > max) then begin
      max:=a[i];
      digmax:=i;
    end;
  DígitoMaximo:=digmax;
end;

```

2. Modifique para informar el dígito que más veces apareció para cada número

Ordenación vectores

CADP 2023



El proceso por el cual, un grupo de elementos puede ser ordenado se conoce como algoritmo de ordenación.

Ordenar

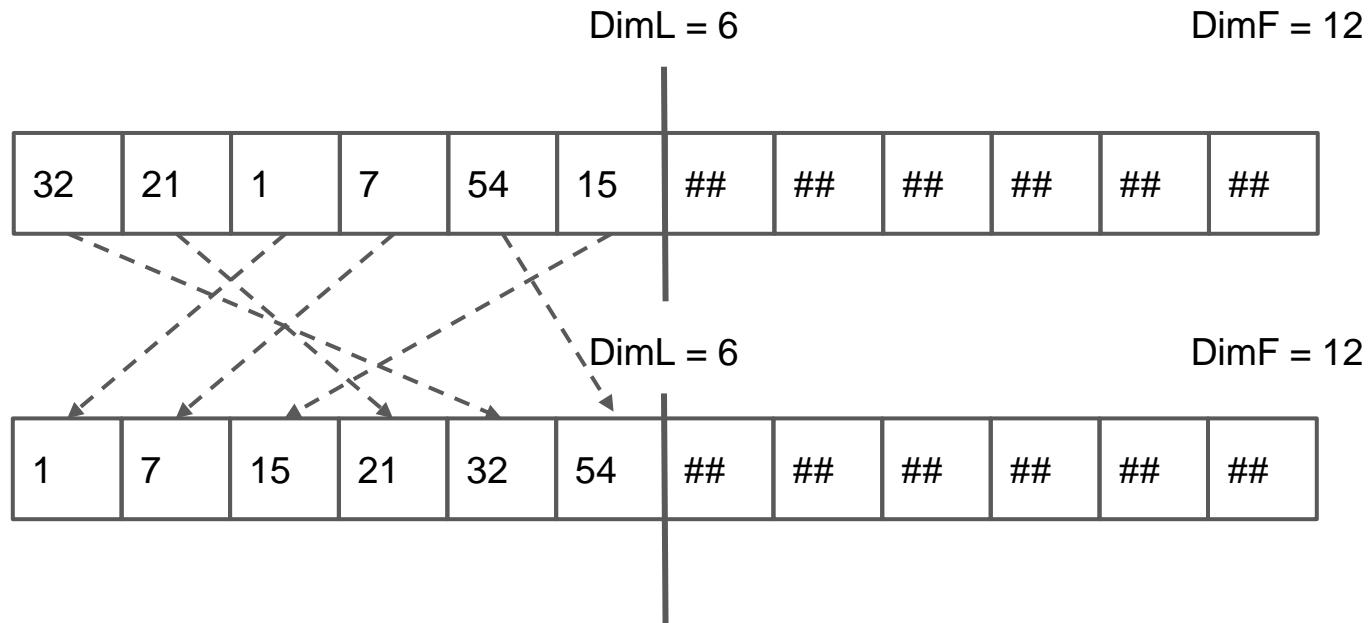
Algoritmos de ordenación

- Selección
- Intercambio
- Inserción

Difieren en:
Dificultad
Memoria
Tiempo

Selección

CADP – Ordenación de vectores - Ejemplo



CADP – Ordenación de vectores - Algoritmo de selección

Cómo funciona?

Es un algoritmo de $\dim L$ pasadas.

Para cada pasada i

Se elige el mínimo en el vector a partir de la posición $(i+1)$ hasta el final

Si el mínimo de vector es más chico que lo que está almacenado en la posición i del vector se intercambia.

CADP – Ordenación de vectores - Algoritmo de selección

dim lógica=6

9	100	85	2	6	150				
---	-----	----	---	---	-----	--	--	--	--

Vector original

dim física = 10

2	100	85	9	6	150				
---	-----	----	---	---	-----	--	--	--	--

Primera pasada

2	6	85	9	100	150				
---	---	----	---	-----	-----	--	--	--	--

Segunda pasada

2	6	9	85	100	150				
---	---	---	----	-----	-----	--	--	--	--

Tercera pasada

2	6	9	85	100	150				
---	---	---	----	-----	-----	--	--	--	--

Cuarta pasada

2	6	9	85	100	150				
---	---	---	----	-----	-----	--	--	--	--

Quinta pasada

CADP – Ordenación de vectores - Algoritmo de selección

```
Program ordenamos;
const
  tam = 150;
type
  numeros= array [1..tam]  of integer;
var
  VN: numeros;  dimL:integer;
begin
  llenarNumeros (VN,dimL);  //ya está hecho
  ordenar(VN,dimL);
end.
```

CADP – Ordenación de vectores - Selección

```
Procedure Ordenar ( var v: numeros; dimLog: integer);
    var i, j, p, item: integer;
    begin
        for i:=1 to dimLog-1 do
            begin {busca el mínimo v[p] entre v[i] , ..., v[N] }
                p := i;
                for j := i+1 to dimLog do
                    if v[ j ] < v[ p ] then
                        p:=j;
                {intercambia v[i] y v[p] }
                item := v[ p ];
                v[ p ] := v[ i ];
                v[ i ] := item;
            end;
        end;
```

Alocación dinámica Punteros

Explicación P5

CADP 2023



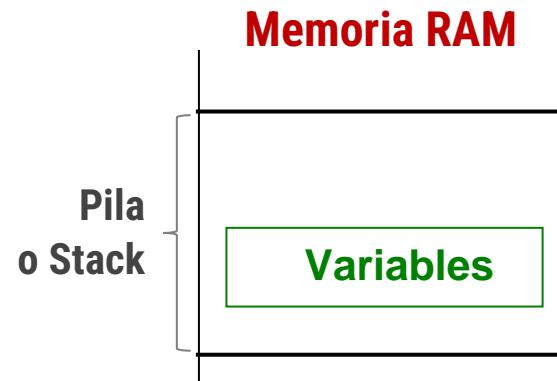
FACULTAD DE INFORMATICA



UNIVERSIDAD
NACIONAL
DE LA PLATA

VARIABLES DINAMICAS

Las variables estáticas ocupan un espacio de memoria que se reserva al declararlas y que no cambia durante la ejecución del programa.



VARIABLES DINAMICAS

Variable Puntero: variable estática que almacena la dirección en memoria de otra variable (*llamada variable dinámica*).

miVariablePuntero

Direcciones de memoria

Memoria

xxxxxxxxxx	
xxxxxxxxxx	
5623	Dato
xxxxxxxxxx	
5623	
xxxxxxxxxx	

Memoria Dinámica

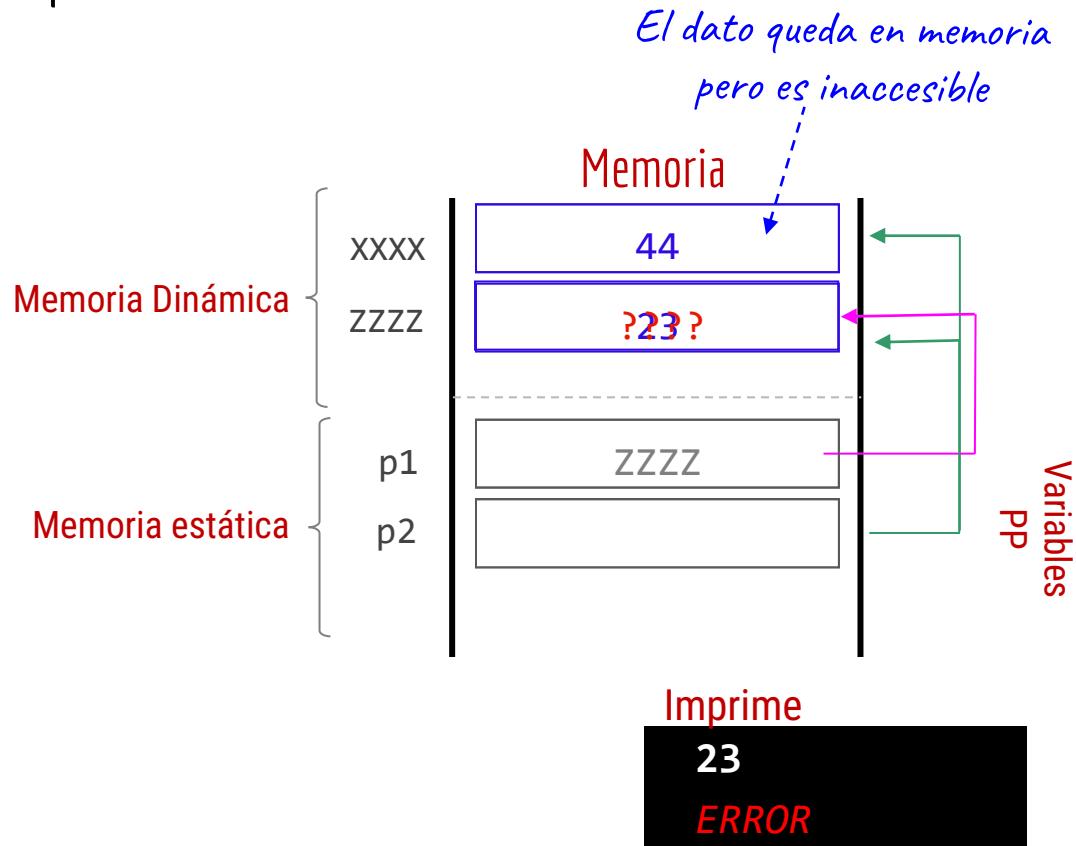
Memoria estática

USO DE PUNTEROS

<p>Declaración (ejemplo)</p> <p>TYPE PunteroEntero = ^ integer;</p> <p>VAR pun, otropun: PunteroEntero;</p>	<p>Valores posibles de un puntero</p> <ul style="list-style-type: none">• NIL• Dirección de memoria dinámica
<p>¿Qué se puede hacer con punteros?</p> <ul style="list-style-type: none">• Reservar memoria dinámica (new)• Liberar memoria dinámica (dispose)• Asignar punteros• Comparar punteros• Acceder al dato apuntado por el puntero (^)	

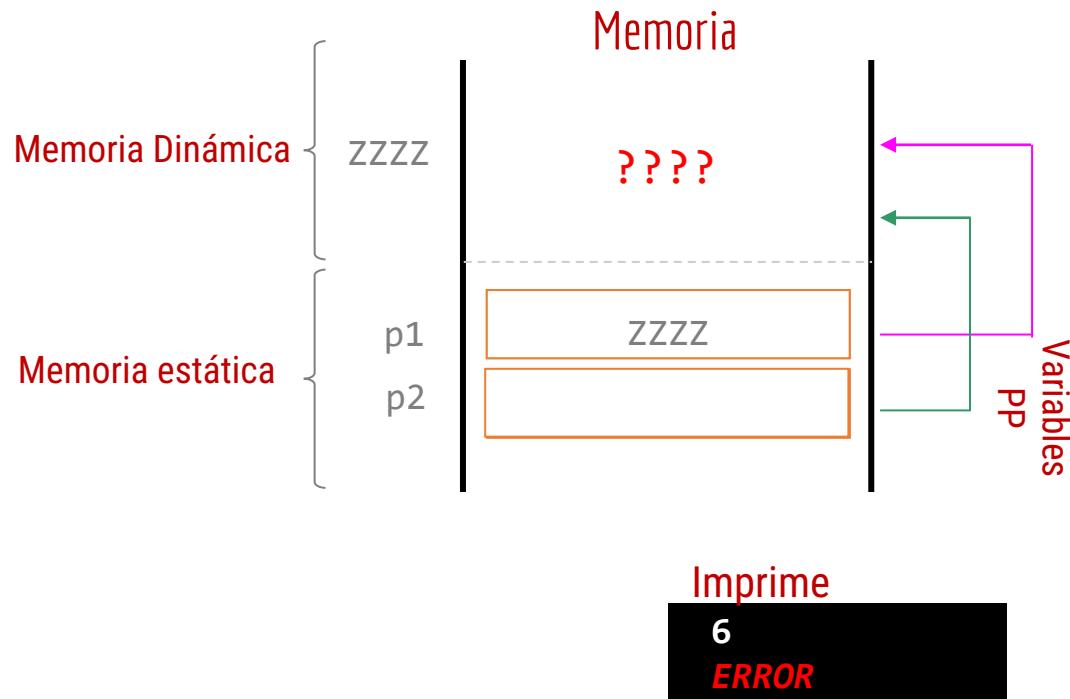
USO DE PUNTEROS - Ejemplo 1

```
Program ejemplo;
Type
  Ptro= ^integer;
Var
  → p1, p2: Ptro;
Begin
  → new (p1);
  → p1^ := 23;
  → new (p2);
  → p2^ := 44;
  → p2 := p1;
  write (p2^);
  dispose (p2);
  → write(p1^);
End.
```



USO DE PUNTEROS - Ejemplo 2

```
Program ejemplo;
Type
  casa = record
    met_cua: real;
    cant_hab: integer;
  end;
  punt_casa = ^casa;
Var
  p1, p2: punt_casa;
Begin
  new (p1);
  p1^.met_cua := 125.50;
  p1^.cant_hab := 5;
  p2:= p1;
  p2^.cant_hab := 6;
  write (p1^.cant_hab);
  dispose (p2);
  write(p1^.met_cua);
End.
```



USO DE PUNTEROS - Ejemplo 3: Punteros como parámetros

```
Program ejemplo;
Type
  punt = ^integer;
```

```
→ Procedure suma (p1:punt);
  Begin
    → p1^ := p1^ + 1;
  End;
```

```
Var
  → p: punt;
```

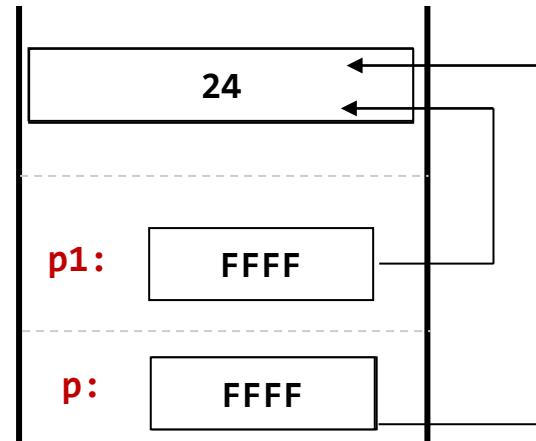
```
Begin
  → new (p);
  → p^ := 23;
  suma(p);
  write(p^);
End.
```

p1 es una copia de p (parámetro por valor). Pero el dato apuntado por ambos es el mismo. Si modifico p1^, también modifico p^

Variables dinámicas

Variables
estáticas del
Proceso suma

Variables estáticas
del Prog.Ppal



Imprime

USO DE PUNTEROS - Ejemplo 4: Punteros como parámetros

```
Program ejemplo;
Type
  punt = ^integer;
```

```
Procedure suma (p1:punt);
Begin
  p1^ := p1^ + 1;
  new(p1);
```

```
End;
```

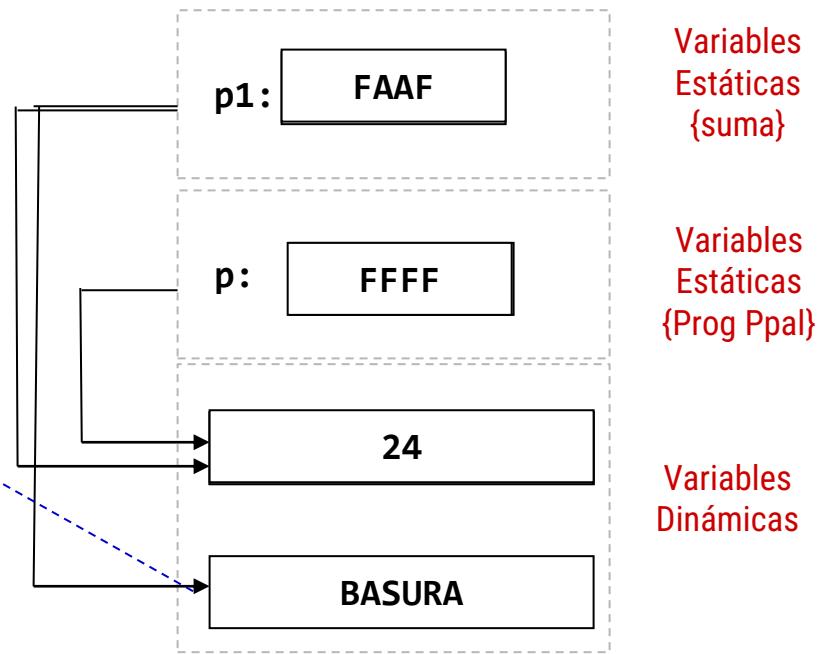
```
Var
  p: punt;
```

```
Begin
  new (p);
  p^ := 23;
  suma(p);
  write(p^);
End.
```

p1 es una copia de p. Si modiflico p1, p en el programa ppal NO va a ver el cambio

¿Qué ocurre al reemplazar new(p1) por dispose(p1)?

El dato queda en memoria pero es inaccesible



IMPRIME 24

USO DE PUNTEROS

Ejemplo 5: Punteros como parámetros

```
Program ejemplo;
Type
  punt = ^integer;

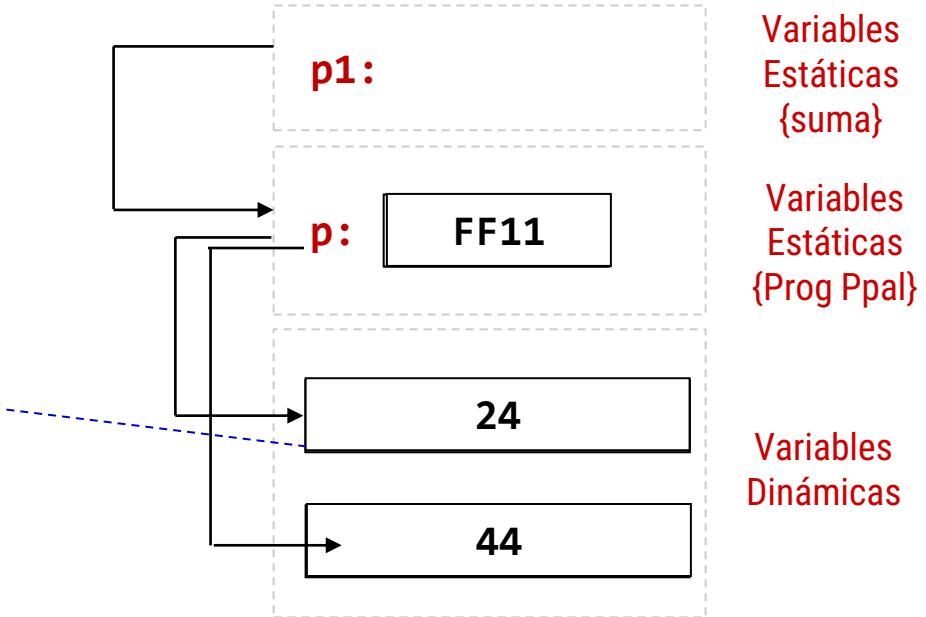
Procedure suma (VAR p1:punt);
Begin
  p1^ := p1^ + 1;
  new(p1);
  p1^:= 44;
End;

Var
  p: punt;

Begin
  new (p);
  p^ := 23;
  suma(p);
  write(p^);
End.
```

p1 recibe la referencia de p. Si modifco p1, estoy modificando p en el programa ppal.

El dato queda en memoria pero es inaccesible



Listas

Explicación P6 - Parte 1

CADP 2023



FACULTAD DE INFORMATICA



UNIVERSIDAD
NACIONAL
DE LA PLATA

Listas - Agregar adelante

Ejercicio 1

Escriba un programa que lea y almacene información de jugadores de básquet. De cada jugador se lee: dni, apellido y nombre, y altura en cm. La lectura finaliza cuando se lee el jugador con dni 0, el cual no debe procesarse.

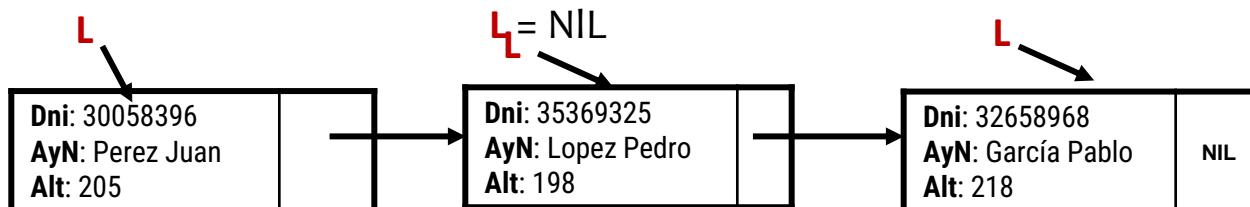
¿Cómo se representa la información de un jugador?

¿En qué estructura de datos se puede almacenar la información de todos los jugadores?



Listas - Agregar adelante

Ejercicio 1



Notar que quedaron almacenados en el orden inverso al leido

Listas - Agregar adelante

```
program ejercicio;
type
  jugador = record
    dni: integer;
    nomyAp: string[30];
    altura: integer;
  end;

  lista = ^nodo;

  nodo = record
    dato: jugador;
    sig : lista;
  end;
```

```
var {PROGRAMA PRINCIPAL}
  L: lista;
begin
  L:= nil;
  cargarLista(L);
end.
```

```
procedure agregarAdelante(var L:lista; j:jugador);
var
  nue: lista;
begin
  new (nue);           {Creo un nodo}
  nue^.dato := j;     {Cargo el dato}
  nue^.sig := L;      {Realizo el enlace}
  L:= nue;             {Actualizo el primero}
end;

procedure cargarLista(var L:lista);
var
  j: jugador;
begin
  leerJugador(j); {Lee un registro de jugador}
  while(j.dni <> 0) do begin
    agregarAdelante(L, j);
    leerJugador(j);
  end;
end;
```

Listas - Recorrido

Ejercicio 2

A partir de la lista generada en el Ejercicio 1, informar la cantidad de jugadores con dni par.



Listas

Explicación P6 - Parte 2

CADP 2023



FACULTAD DE INFORMATICA



UNIVERSIDAD
NACIONAL
DE LA PLATA

Listas - Agregar atrás

Ejercicio 1

Escriba un programa que lea y almacene información de jugadores de básquet. De cada jugador se lee: dni, apellido y nombre, y altura en cm. La lectura finaliza cuando se lee el jugador con dni 0, el cual no debe procesarse. La información de los jugadores debe quedar almacenada en el mismo orden en que fue leída.

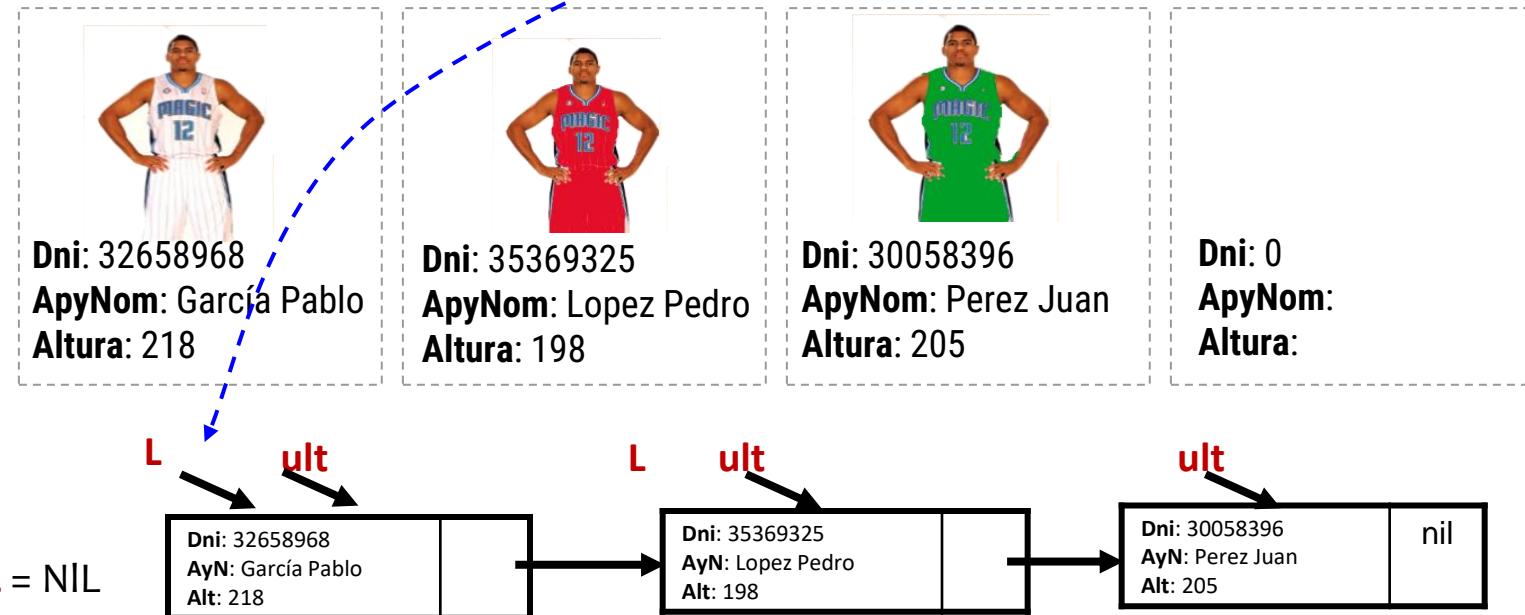
¿Cómo debería realizarse la carga?



Listas - Agregar atrás

El puntero al primer nodo de la lista se mantiene fijo y el puntero al último se va actualizando

Ejercicio 1



Notar que quedaron almacenados en el orden leido

Listas - Agregar atrás

```
program ejercicio;
type
  jugador = record
    dni: integer;
    nomyAp: string[30];
    altura: integer;
  end;

  lista = ^nodo;

  nodo = record
    dato: jugador;
    sig : lista;
  end;
```

```
var {PROGRAMA PRINCIPAL}
  L: lista;
begin
  L:= nil;
  cargarLista(L);
end.
```

```
ULT: lista;
Begin
  leerJugador(j);

  while(j.dni <> 0) do begin
    agregarAtras(L, ULT, j);
    leerJugador(j);
  end;
end;

procedure agregarAtras(var L, ULT:lista;
  j:jugador);
var
  nue: lista;
begin
  new (nue); {Creo un nodo}
  nue^.dato := j; {Cargo el dato}
  nue^.sig := nil; {Inicializo
    enlace en nil}
  if( L = nil) then {Si la lista está vacía}
    L:= nue {Actualizo el
    inicio}
  else {Si la lista no
    está vacía}
    ULT^.sig := nue; {Realizo enlace con el
    último}
  ULT := nue; {Actualizo el
    último}
```

Listas

Explicación P6 - Parte 3

CADP 2023



FACULTAD DE INFORMATICA



UNIVERSIDAD
NACIONAL
DE LA PLATA

Listas - Insertar ordenado

Ejercicio 2

Modifique la solución del ejercicio 1 para que la información de los jugadores quede ordenada **por altura** de manera ascendente.

¿Cómo debería realizarse la carga?



Listas - Insertar ordenado

La forma de realizar los enlaces varía de acuerdo al lugar donde corresponde insertar el nodo

Ejercicio 2



Dni: 32658968
ApyNom: García Pablo
Altura: 218



Dni: 35369325
ApyNom: Lopez Pedro
Altura: 198

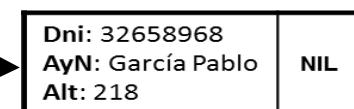
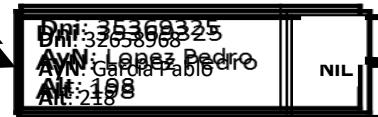


Dni: 30058396
ApyNom: Perez Juan
Altura: 205

Dni: 0
ApyNom:
Altura:

L = NIL

L **L**



Notar que quedaron almacenados ordenados por altura de forma ascendente

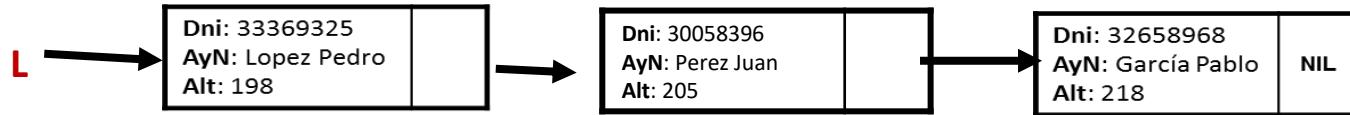
Listas - Insertar ordenado

Casos:

1- Insertar nodo en una lista vacía



2- Insertar en lista no vacía



Pueden darse 3 casos: insertar adelante - insertar en el medio - insertar atrás

Al inicio

Dni: 34769325 AyN: Fernández Juan Alt: 185	nil
--	-----

Al medio

Dni: 29769325 AyN: Martínez Carlos Alt: 210	nil
---	-----

Al final

Dni: 32769325 AyN: González Mario Alt: 219	nil
--	-----

Listas - Insertar ordenado

Pasos a seguir:

1.Crear el nodo a insertar

2.Buscar la posición correspondiente para insertar el nodo creado

- *Es necesario ubicarse al inicio de la lista y recorrer hasta encontrar la posición.*
- *Vamos a utilizar 2 punteros auxiliares para realizar el recorrido. ¿por qué?*

3.Realizar los enlaces

- *Una vez encontrada la posición, deberán actualizarse los enlaces de acuerdo al caso. Es importante entonces determinar dicho caso*

Listas - Insertar ordenado

```
procedure cargarLista(var L:lista);
var
  j: jugador;
Begin
  leerJugador(j);
  while(j.dni <> 0) do begin
    insertarOrdenado(L, j);
    leerJugador(j);
  end;
end;
```

1. Crear el nodo a insertar
2. Buscar la posición para insertar el nodo creado
3. Realizar los enlaces

```
procedure insertarOrdenado(var L:lista; j:jugador);
var
  nue: lista;
  act, ant: lista; {punteros auxiliares para recorrido}
begin
  new (nue);
  nue^.dato := j;
  act := L; {ubico act y ant al inicio de la lista}
  ant := L;
  while( act <> nil)and(j.altura > act^.dato.altura)do
  begin
    ant := act;
    act:= act^.sig;
  end;
  if (act = ant) then {al inicio o Lista vacía}
    L:= nue;
  else {al medio o al final}
    ant^.sig:= nue;
    nue^.sig:= act;
  end;
```