

# Resumen teoria Final/promocion

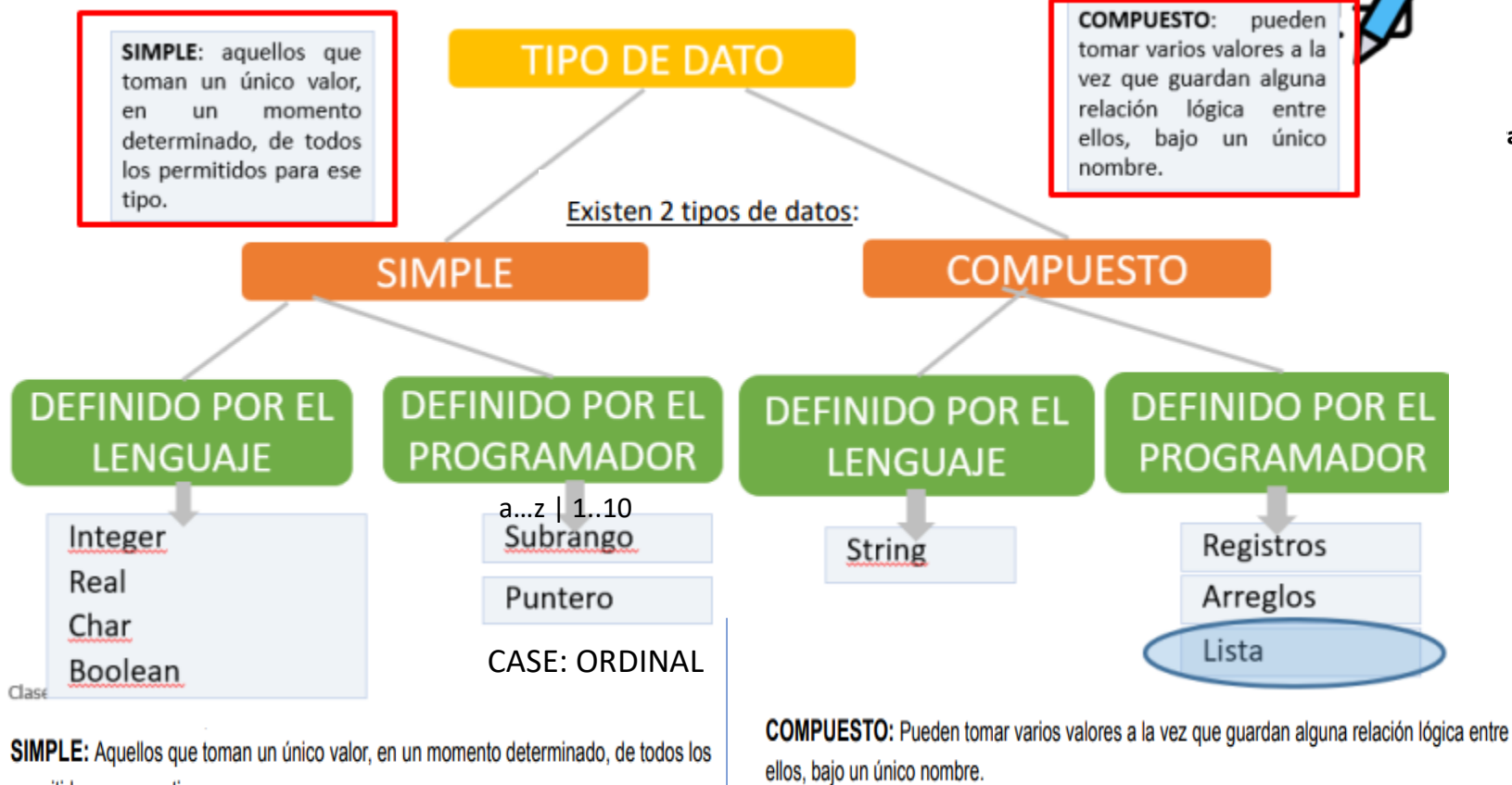
- Respuestas cortas y directamente que conteste la pregunta

-

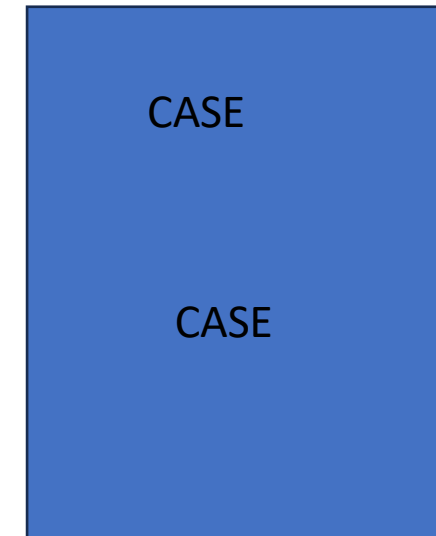
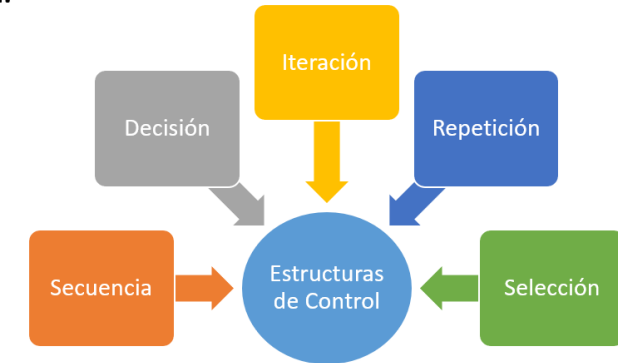


El Final: [Link](#)

# Tipos de datos



acción.



Funcion: solo retorna,  
datos de tipo simple

# Estructuras de Control

Secuencia

Decisión

If / Else

Iteración

While | Repeat

Repetición

For

Selección

Case: simple y ordinal Y disjunta

- **Iteración:** puede ocurrir que se desee ejecutar un bloque de instrucciones desconociendo el número exacto de veces que se ejecutan. Para estos casos existen en la mayoría de los lenguajes de programación estructurada

Existen 2 tipos:

- **Precondicionales:** evalúan la condición y si es verdadera se ejecuta el bloque de acciones. Se puede ejecutar varias veces. El valor de la condición debe ser evaluable antes de realizar la evaluación de la condición.  
Por ejemplo: While.
- **Postcondicionales:** ejecutan las acciones, luego evalúan la condición y ejecutan las acciones mientras la condición sea falsa.  
Por ejemplo: Repeat Until

- **Decisión:** en un algoritmo representativo de un problema real es prácticamente imposible que las instrucciones sean secuenciales puras. Es necesario tomar decisiones en función de los datos del problema. Por ejemplo: If.

- **Repetición:** consiste en repetir N (deben conocerse de antemano) veces un bloque de acciones. Por ejemplo: for.

- **Selección:** permiten realizar distintas acciones dependiendo del valor de una variable de tipo **ordinal**. Por ejemplo: case.

La estructura de control más simple, está representada por una sucesión de operaciones (por ej. asignaciones), en la que el orden de ejecución coincide con el orden físico de aparición de las instrucciones.

```
Programa uno;
...
var
  num: integer;
begin
  read (num);
  write (num);
end.
```

- Representada por una sucesión de operaciones (por ej. asignaciones), en la que el orden de ejecución coincide con el orden físico de aparición de las instrucciones.
- **Secuencia:** estructura de control más simple está representada por una sucesión de operación (por ejemplo asignación), en la que el orden de ejecución coincide con el orden físico de aparición de las instrucciones.

## Pre-Condicionales(While)

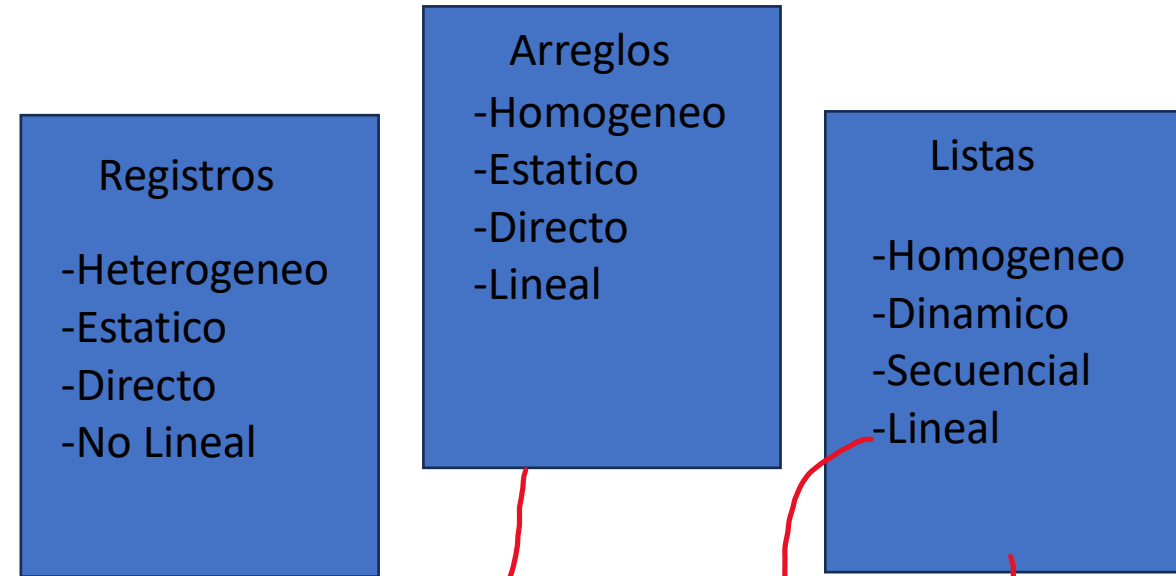
Evalúan la condición y si es verdadera se ejecuta el bloque de acciones. **Dicho bloque se pueda ejecutar 0, 1 ó más veces.** Se ejecuta mientras sea verdadera. (El valor inicial de la condición debe ser conocido o evaluable antes de la evaluación de la condición)

## Post-Condicionales(Repeat-Until)

Ejecutan las acciones luego evalúan la condición y ejecutan las acciones mientras la condición es falsa. **Puede ejecutarse 1 o más veces.**

# Estructuras de Datos

- **Elementos:** de que tipo son sus elementos
  - Homogénea: elementos del mismo tipo
  - Heterogénea: elementos de distinto tipo
- **Tamaño:** si puede variar su tamaño a lo largo del programa
  - Estática: el tamaño no varía
  - Dinámica: el tamaño varia
- **Acceso:** como se puede acceder a sus elementos
  - Secuencial: Para acceder a un elemento particular se debe respetar un orden predeterminado
  - Directo: Se puede acceder a un elemento particular, directamente, sin necesidad de pasar por los anteriores a él
- **Linealidad:** como están almacenados los elementos que la componen
  - Lineal: guardan una relación de adyacencia ordenada donde a cada elemento le sigue uno y le precede uno, solamente.
  - No Lineal: Para un elemento dado pueden existir 0, 1 ó más elementos que le suceden y 0, 1 ó más elementos que le preceden.



Logico  
no Por direcciones

## Clasificación:

1. Vectores: tipo de dato arreglo con un solo índice (sólo una dimensión)
2. Matrices: tipo de dato arreglo con 2 índices (2 dimensiones)
3. Tensores

## Ocupación de memoria

- En su declaración siempre va a ser de 4 bytes(en la memoria estática)
- En su inicialización ocupa sus 4 bytes(del puntero), más el contenido que tiene (en la memoria dinámica)

# Alcance de Variables

Program alcance;

Var

a,b: integer; → **Variables Globales del Programa** (Pueden ser usadas en todo el programa (incluyendo módulos))

procedure prueba;

Var

c: integer; → **Variable Local** (Pueden ser usadas sólo en el proceso que están declaradas)

Begin

End.

Var

d:integer; → **Variable Local del Programa** (Pueden ser usadas sólo en el cuerpo del programa)

Begin

End.

Si es una variable utilizada en un proceso:

- Se busca si es variable local
- Se busca si es un parámetro
- Se busca si es variable global al programa

Si es una variable usada en un programa:

- Se busca si es variable local al programa
- Se busca si es variable global al programa

## Clasificación de ESTRUCTURAS DE DATOS

- **Elementos:** de que tipo son sus elementos
  - Homogénea: elementos del mismo tipo
  - Heterogénea: elementos de distinto tipo
- **Tamaño:** si puede variar su tamaño a lo largo del programa
  - Estática: el tamaño no varía
  - Dinámica: el tamaño varia
- **Acceso:** como se puede acceder a sus elementos
  - Secuencial: Para acceder a un elemento particular se debe respetar un orden predeterminado
  - Directo: Se puede acceder a un elemento particular, directamente, sin necesidad de pasar por los anteriores a él
- **Linealidad:** como están almacenados los elementos que la componen
  - Lineal: guardan una relación de adyacencia ordenada donde a cada elemento le sigue uno y le precede uno, solamente.
  - No Lineal: Para un elemento dado pueden existir 0, 1 ó más elementos que le suceden y 0, 1 ó más elementos que le preceden.



# Alocacion de Memoria

```
Type
vector = array[1..5] of real;
Var
v:vector;
letra:char;
num:integer;
ok:boolean;
p:punteroAEntero; //ya veremos como
```

Al comenzar mi programa ocupa

$v = 5 * 8 = 40$  bytes

letra = 1 byte

num = 6 bytes

ok = 1 byte

p = 4 bytes

**52 bytes de memoria  
estática**

**DISPOSE (p)**

Libera la conexión que existe entre la variable y la posición de memoria.

Libera la posición de memoria.

La memoria liberada puede utilizarse en otro momento del programa.

**p:=nil**

Libera la conexión que existe entre la variable y la posición de memoria.

La memoria sigue ocupada.

La memoria no se puede referenciar ni utilizar.



MEMORIA  
DINAMICA

MEMORIA  
ESTATICA

Si durante la ejecución del programa p **reserva** memoria se ocuparán tantos bytes de memoria dinámica como sea el contenido de p (en este caso 6 bytes de memoria dinámica). Luego p podrá **liberar** esa memoria dinámica durante la ejecución del programa.

Tipo de variable	Bytes que ocupa
Char	1 byte
Boolean	1 byte
Integer	6 bytes
Real	8 bytes
String	Tamaño + 1
Subrango	Depende el tipo
Registro	La suma de sus campos
Vector	Dimensión física * tipo elemento
Puntero	4 bytes

# Calculos de mem

## Estatica / Dinamica

### Estatica

Que miramos?

- Constantes
- variables del program principal
- Variables locales a los modulos
- Parametros
- Variables Globales

### Dinamica

Que miramos?

- Los New
- Los dispose
- (Los nil no, solo corta enlaces)
- Parametros

### Operaciones

Vectores =  $\text{dimF} * \text{tipo de dato}$

Registros = la suma de sus campos

String =  $\text{cant} + 1$

Subrangos (1..4) | (a..z) = tipo de Dato(porq se va elegir 1 solo)

# Calculos de mem

## Estatica / Dinamica

EstaticaQue miramos?

- Constantes
- variables del program principal
- Variables locales a los modulos
- Parametros
- Variables Globales

```

Type temperaturas = array [1..30] of real

Function contar ( tem:temperaturas): integer
Var i: 1..30; can10 : integer;
begin
  can10 := 0;
  {recorrido total del vector}
  For i := 1 to 30 do
    If (tem[i] = 10) and (i mod 2 = 0)
    then can10 := can10 + 1;
  contar := can10;
end.

```

Handwritten annotations:

- 30 x  $\mathbb{R}$  (pointing to the array type)
- 30 x ENTERO (pointing to the loop range)
- 1 x ENTERO (pointing to the can10 variable)

Tabla de ocupación:

char,	(1 byte)
boolean,	(1 byte)
integer	(4 bytes)
real,	(8 bytes)
string,	(tamaño + 1 byte)
subrango,	(depende el tipo)
registro,	(suma de sus campos)
arreglos	(dimFisica*tipo e
puntero	(4 bytes)

Programa uno;

```

Type
  puntero = ^real;
  puntero2 = ^char;
  persona = record
    nombre:string[20];
    dni:integer;
  end;
  punPer = ^persona;
Var
  p,p1:puntero;
  per: punPer;
Begin
  new(per);
  new(p);
  p^:= 8;
  p1:= p;
  dispose(p1);
End.

```

Handwritten annotations:

- 12 (pointing to the program name)
- 8 (pointing to the p, p1: puntero; line)
- 6 (pointing to the per: punPer; line)
- Dinamica (written in blue)
- 27 + 9 = 25 (boxed, pointing to the new(per); and new(p); lines)
- 8 (pointing to the p1:= p; line)
- p = [ABC] (pointing to the p1:= p; line)
- p1 = [ ] (pointing to the dispose(p1); line)

Que miramos?

- Los New
- Los dispose
- (Los nil no, solo corta enlaces)
- Parametros



# Calculos de tiempos de ejecucion en mem y eficiencia

## Tiempo Del Algoritmo

- Existen algoritmos que el tiempo de ejecución depende de la cantidad de datos de entrada o su tamaño
- Existen otros algoritmos donde el tiempo de ejecución es una función de la entrada "específica"(se elige el peor caso)

**Análisis Empírico:** Se escribe el código, se ejecuta en una máquina y se mide el tiempo

- Ventaja:
  - El tiempo es preciso
  - Fácil de realizar
- Desventaja
  - Tengo que escribir el programa primero
  - Obtiene valores exactos para una máquina y unos datos determinados (dependiente de la máquina donde se ejecute)
  - Requiere ejecutarlo varias veces

**Análisis Teórico:** Se hace un análisis sobre algunas de las instrucciones que hace el programa y se estima un tiempo que el algoritmo necesita para su ejecución, **no se necesita ejecutarlo.**

- Ventajas
  - Obtiene valores aproximados
  - Es aplicable en la etapa de diseño de los algoritmos (se puede aplicar sin necesidad de implementar el algoritmo)
  - El análisis es independiente de la máquina donde se ejecute

### REGLAS GENERALES

1. **Sentencias Consecutivas:** maximo de todas las instrucciones  
 $\max(\text{inst1}, \text{inst2})$
  2. **For / For Anidados:** Se debe calcular la cantidad de operaciones elementales que se ejecutan dentro del FOR y multiplicarla por la cantidad de veces que se ejecuta la instrucción FOR.
  3. **While / Repeat...Until:** Se debe calcular la cantidad de operaciones elementales que se ejecutan dentro del WHILE y multiplicarla por la cantidad de veces que se ejecuta el WHILE. Como no se conoce esa cantidad se considera el PEOR CASO. Por ejemplo, se supone una cantidad
- If / Else:** En el caso de una sentencia IF en su forma completa (then/else), debe calcularse la cantidad de operaciones que se realizan en cada parte y se debe elegir aquella que consuma más tiempo  
 $\max(\text{If}, \text{Else})$

Que miramos?

- Operaciones aritmetica logicas
- Asignaciones
- Comparaciones

Que No miramos

- Writeln
- Readln
- New
- Dispose

### FORMULAS PARA MEDIR TIEMPO DE EJECUCION

FOR  $\rightarrow (OP * m) + (3 * m + 2)$

WHILE  $\rightarrow (m * \text{condV}) + \text{condF} + (m * \text{cuerpo})$

REPEAT  $\rightarrow (m * \text{condV}) + \text{condF} + (m * \text{cuerpo}) + \text{cuerpo}$

$$\begin{array}{r}
 25 \\
 2 \\
 1 \\
 \hline
 2807
 \end{array}$$

IF/ELSE se toma el peor caso

 $25 \sqrt{25}$ 

Repeat: **CONDICIONES**(CANT\_ITERACIONES) + CANT\_ITERACIONES(**CUERPO**)

```

PROCEDURE EJEMPLO();
begin
    i := 0;
    repeat
        begin
            i := i + 1;
        end
    until i = 3;
end;

```

1Ut

3\*3=9Ut

3\*6=18Ut

28Ut

## Que No miremos

- WriteIn
- ReadIn
- New
- Dispose

# Preguntas Tipicas

Si un IF puede reemplazar siempre a un case?

Rt: no puede el case permite datos ordinales (ejemplo no permite preguntar por un string o real)

En Funcion de N, en cálculos de tiempo | el peor caso en memoria

Que que ocupa mas A: un boolean pasado por valor + char por valor O B: un char pasado por referencia y un boolean por referencia?

Rta

que es mas eficiente una Lista pasada por parametro o un vector pasado por referencia?

Los valen lo mismo, por q los 2 son tipo puntero, ninguno es mas o menos eficiente (cuando se pasa por referencia vale 4)

Saben si es siempre posible eliminar el primer elemento de una lista?

Si no mal recuerdo al igual que el vector eso es falso, ya que podría no existir dicho primer elemento y la lista estar vacía

15:32

Disjunta: ejemplo en un case, una variable caracter va ser Minuscula o mayuscula no las 2 cosas

```
begin
1) dimL:=5; -1
   read(a.ape_nom);
2) i:=0; -1
   while (i < 5) and (a.ape_nom <> 'ZZZ') do
   begin
       read(a.promedio);
       new (v[i]);
       v[i]^:= a;
       i:= i+1;
       read(a.ape_nom);
   end;
end.
```

Handwritten notes:  $5 \times 5 = 270$  Dija

Ordinal: tipo de dato que se puede comparar y preguntar si es mayor que otro (conocer anterior/o el q le sigue)

# Preguntas Tipicas

Si un IF puede reemplazar siempre a un case  
 Rt: no puede el case permite datos ordinales

Que que ocupa mas A: un boolean pasado por valor  
 Rta

que es mas eficiente una Lista pasada por parametro  
 Los valen lo mismo, por q los 2 son tipo puntero

Saben si es siempre posible eliminar el primer elemento de una lista?

Si no mal recuerdo al igual que el vector eso es falso, ya que podría no existir dicho primer elemento y la lista estar vacía

15:32

Disjunta: ejemplo en un case, una variable con  
 o mayuscula no las 2 cosas

Ordinal: tipo de dato que se puede comparar y

```

8
9
10 program Hello;
11
12
13     procedure sarasa(a: integer; var b: real);
14     begin
15         Writeln('aa ',a);
16         Writeln('bb ',b:2:2);
17     end;
18 begin
19     writeln ('Hello World');
20     sarasa(2,8);
21 end.

```

F

<

input


Compilation failed due to following error(s).

```

Free Pascal Compiler version 3.2.2+dfsg-9ubuntu1 [2022/04/11] for x86_64
Copyright (c) 1993-2021 by Florian Klaempfl and others
Target OS: Linux for x86-64
Compiling main.pas
main.pas(20,13) Error: Variable identifier expected
main.pas(23) Fatal: There were 1 errors compiling module, stopping
Fatal: Compilation aborted
Error: /usr/bin/ppcx64 returned an error exitcode

```

# Preguntas Tipicas



En Pascal, los tipos personalizados se pueden declarar dentro de un procedimiento, función o incluso dentro de un bloque begin...end dentro de un procedimiento o función. Sin embargo, ten en cuenta que los tipos declarados dentro de un procedimiento solo serán visibles y accesibles dentro de ese mismo procedimiento o función y sus subprocedimientos o subfunciones. Estos tipos se consideran locales al ámbito del procedimiento o función en el que se declaran.

Aquí tienes un ejemplo de cómo declarar un tipo personalizado dentro de un procedimiento en Pascal:

```
pascal
```

Copy code

```
8
9
10 program Hello;
11
12
13     procedure sarasa(a: integer; var b: real);
14     begin
15         Writeln('aa ',a);
16         Writeln('bb ',b:2:2);
17     end;
18 begin
19     writeln ('Hello World');
20     sarasa(2,8);
21 end.
```

input

Compilation failed due to following error(s).

```
Free Pascal Compiler version 3.2.2+dfsg-9ubuntu1 [2022/04/11] for x86_64
Copyright (c) 1993-2021 by Florian Klaempfl and others
Target OS: Linux for x86-64
Compiling main.pas
main.pas(20,13) Error: Variable identifier expected
main.pas(23) Fatal: There were 1 errors compiling module, stopping
Fatal: Compilation aborted
Error: /usr/bin/ppcx64 returned an error exitcode
```



# Preguntas Tipicas

github.com/NahuelArn

Si un IF puede reemplazar siempre a un case?

Rt: no puede el case permite datos ordinales (ejemplo no permite preguntar por un string o real)

Que que ocupa mas A: un boolean pasado por valor + char por valor O B: un char pasado por referencia y un boolean por referencia?

Rta

que es mas eficiente una Lista pasada por parametro o un vector pasado por referencia?

Los valen lo mismo, por q los 2 son tipo puntero, ninguno es mas o menos eficiente (cuando se pasa por referencia vale 4)

Saben si es siempre posible eliminar el primer elemento de una lista?

Si no mal recuerdo al igual que el vector eso es falso, ya que podría no existir dicho primer elemento y la lista estar vacía

15:32

Disjunta: ejemplo en un case, una variable caracter va ser Minuscula o mayuscula no las 2 cosas

Ordinal: tipo de dato que se puede comparar y preguntar si es mayor que otro (conocer anterior/o el q le sigue)

# Preguntas Tipicas

**Un programa es correcto si se realiza de acuerdo a sus especificaciones.**

**Técnicas para  
corrección de  
programas**

