TEMA VI:

Iteración
- a) Identificar
  - acciones (leer)
  - variables (Dispara / No dispara)
- b) Actualizar variables (tiende a un fin).
- c) Condiciones terminación.
- d) Valores iniciales.

↓

VERIFICACIÓN FORMAL

{P ≡ Invariante} Repite M.q. ( ) {P}

≡

}

VERIFICACIÓN PROG

CORRECIÓN → Lógica Pre-Post (predicados)

↓

ESTILO

↓

EFICICENCIA

{Condiciones entrada → Prec.}

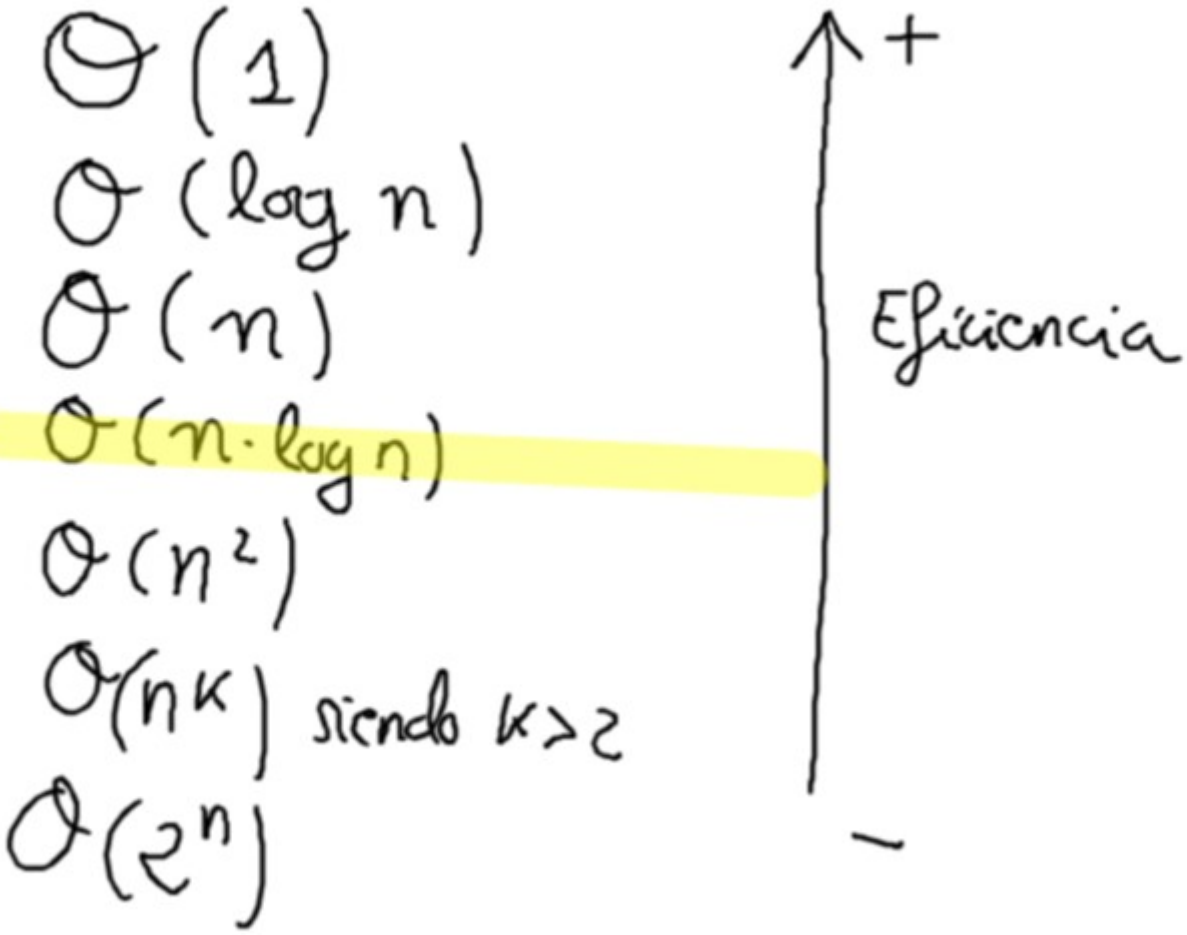
{P1}
{P2} } Algoritmo
{P3}

{Condiciones salida → Post}

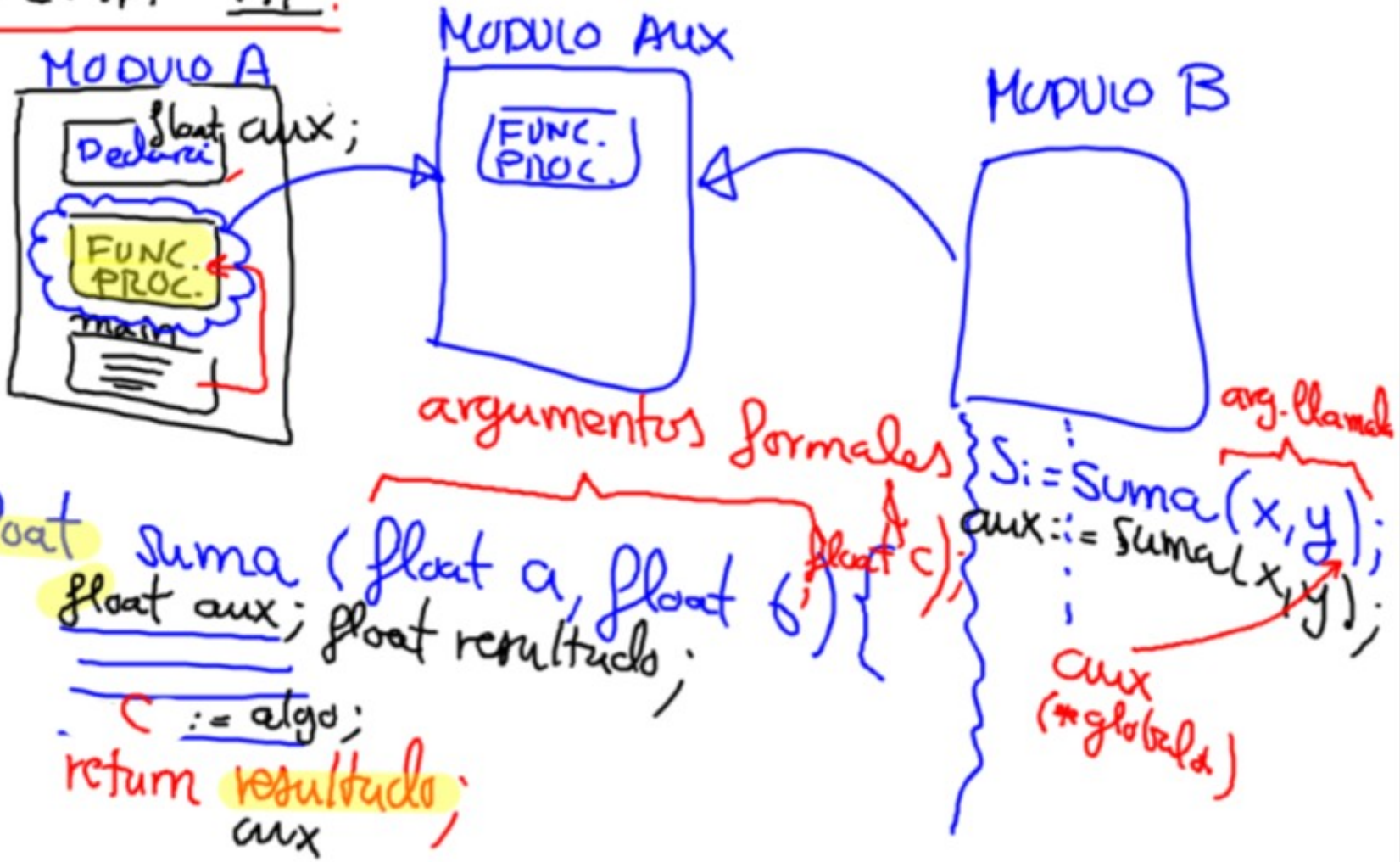
- Consumir Pocos Recursos
  - tiempo ~ tamaño Problema (n.º instrucciones)
  - Memoria
- Análisis
  - tiempo/instrucción (complejidad)
  - n.º instrucciones de cada clase

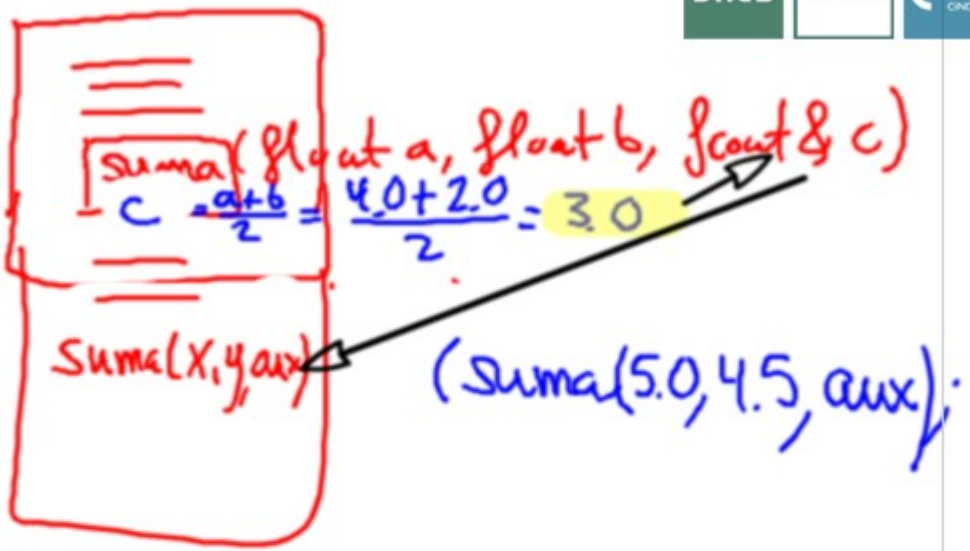
S = a + b

TIPOS DE COMPLEJIDAD.



TEMA VII:


MODULO A
float aux;
Declara
FUNC. PROC.
main
MODULO AUX
FUNC. PROC.
MODULO B
argumentos formales
arg. llamada
float suma (float a, float b, float c);
float aux; float resultado;
C := algo;
return resultado;
aux
S:= Suma (x, y);
aux:= suma(x, y);
aux
(*global)




```
main ( ) {
  float r;
  int fac_G, n, x, y;
  fac_G = factorial (n);
  n = cambiar(x,y);
  cambiar(x,y);
```

1.3. a) (35.3 x 5.1) + (4.5/7.6)

b) ((Float(25)/(-6.2)) x 54)/2.4

c) 25.5 x int(3.5) → ¡OJO!
NO CUMPLE MANUAL ESTILO

d) ((334/6)%4)-(5x4)

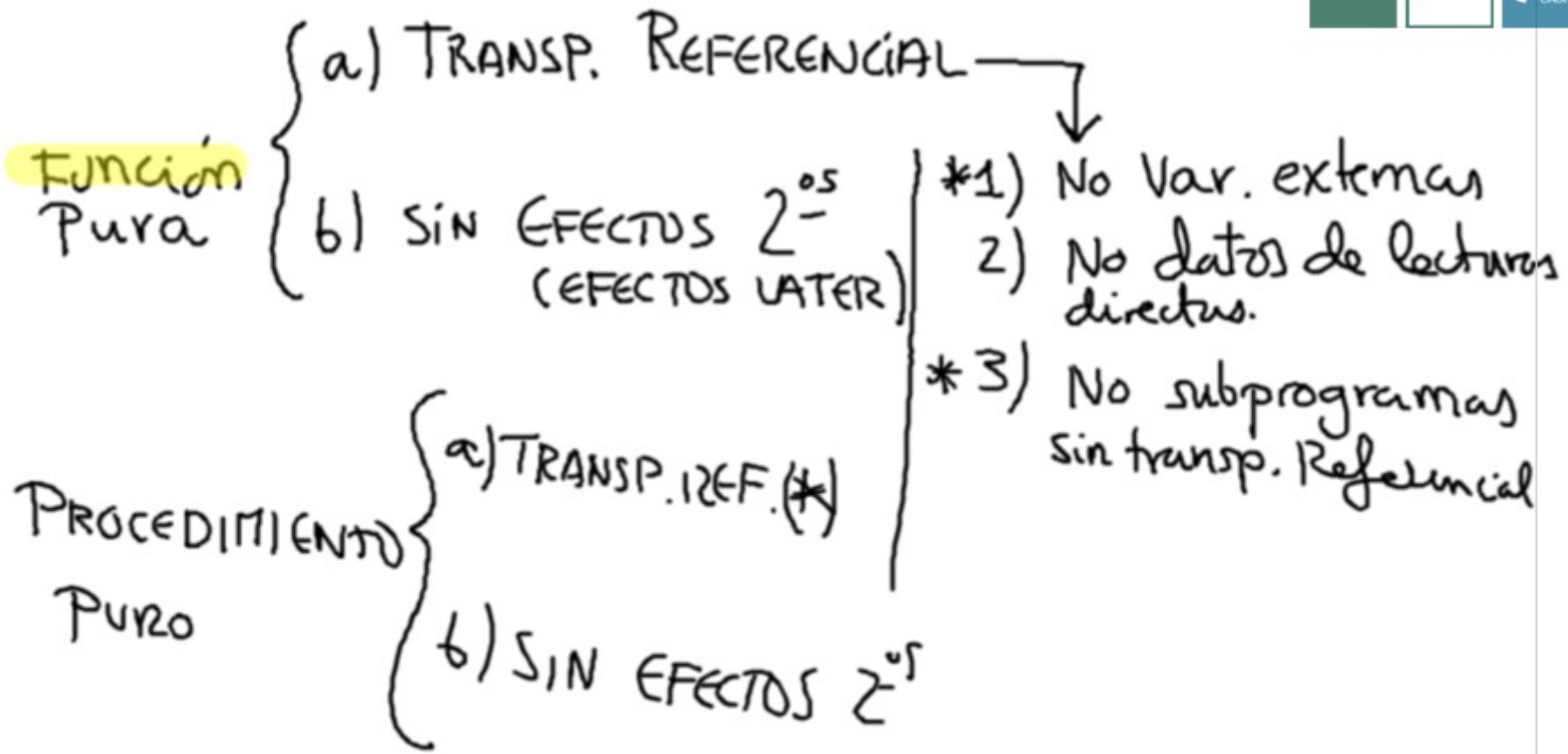
CASA ≠ Casa ≠ casa

TEMA VIII

Abstracción (subprogramas)
- especificación → QUE HACE
- Implementación → COMO lo HACE

Cabecera subprograma + comentario

```
float mysqrt (float n);
```


Función Pura
a) TRANSP. REFERENCIAL
b) SIN EFECTOS 2os (EFECTOS LATER)
*1) No Var. externas
2) No datos de lecturas directas.
*3) No subprogramas sin transp. Referencial
PROCEDIMIENTO PURO
a) TRANSP. REF. (*)
b) SIN EFECTOS 2os

DESARROLLO REFIN. SUCCESIVOS (ABSTRACIÓN)

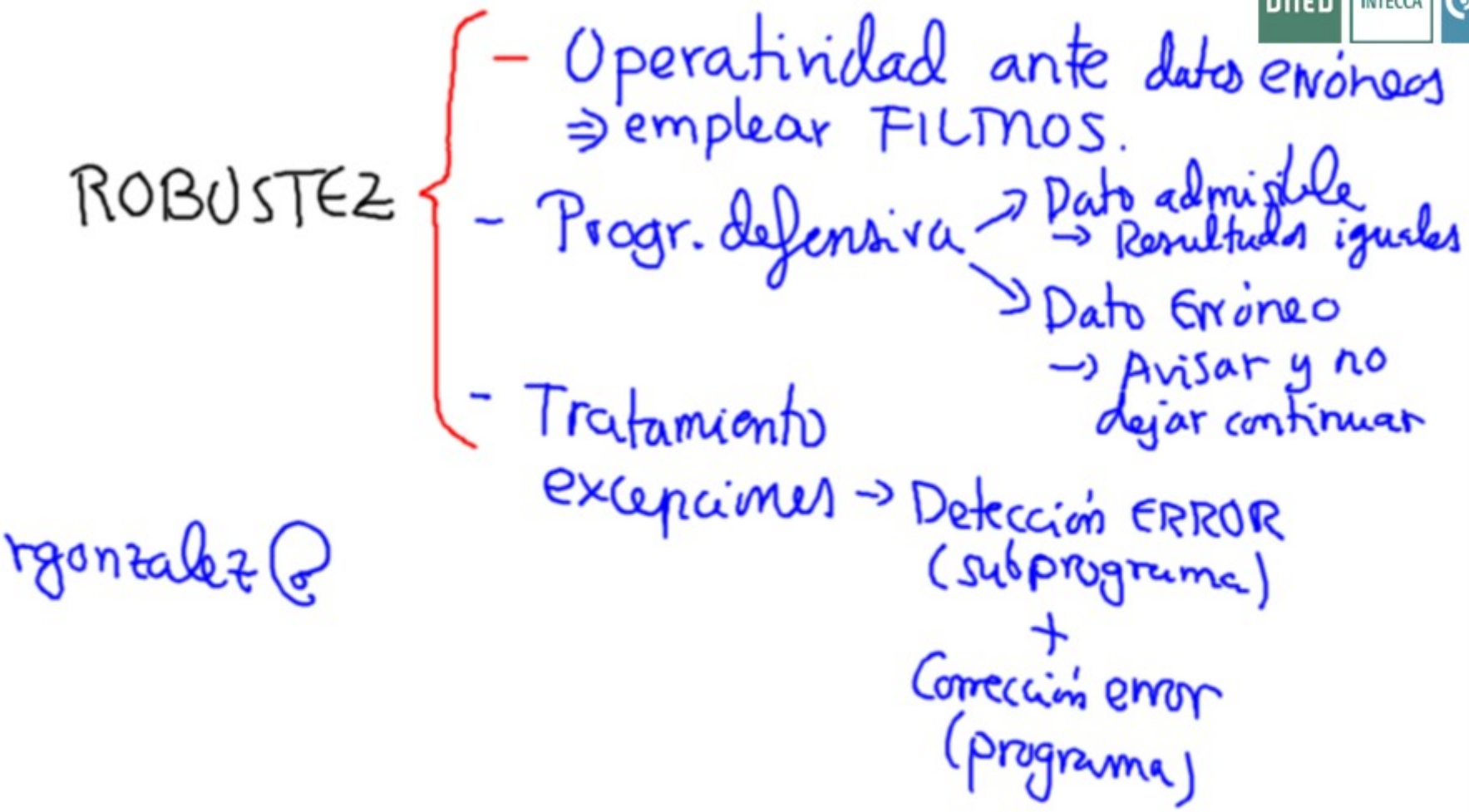
1) ASCENDENTE
   - Desarrollo detalles
   - especificación general

2) DESCENDENTE
   - Especificamos general
   - Desarrollo de detalle

3) REUTILIZACIÓN
   - Subprogramas lo más aprovechables
     V: Campo de aplicación máximo.
     I: Coste de desarrollo.

VENTAJAS
- Reutilizac
- Claridad
- Organizad
- Eficiencia memoria

INCONVENIENTES
- Eficiencia ejecución.


ROBUSTEZ
- Operatividad ante datos erróneos
⇒ emplear FILTROS.
- Progr. defensiva
Dato admisible → Resultados iguales
Dato Erróneo → Avisar y no dejar continuar
- Tratamiento excepciones → Detección ERROR (subprograma) + Corrección error (programa)

rgonzalez

```
for (int cont1 = -1; cont1 <=10;
        cont1++) {
    for (int cont2 = 1 ;cont2 <= cont1; cont2++){
        printf ("Hola\n");
    }
}
```

```
int n=0;
if(n<2){
    n++;
}
```

```
typedef enum DiaSemana {
        Lunes, Martes, .... } ;
```

0 1

ordinales

```
DiaSemana MiSemana;
MiSemana = DiaSemana{ };
MiSemana = Lunes;
if (MiSemana ≥ Lunes) {

}
```

int (Diciembre) == 11

~~Diciembre = 11~~

MisMeses = Meses (11);

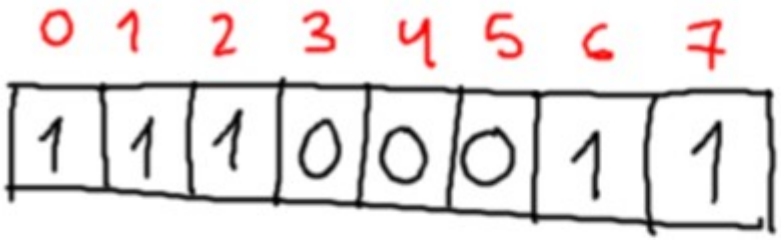
MisMeses = Meses ((int)(MisMeses) - 1);

mes 10 == noviembre.

VECTOR:

indice

Datos (cualquier_tipo)




```
typedef Tipo_V [n];
```

(tipo_Datos)

```
typedef bool tipo_juego [8];
  :
tipo_juego mi_juego = {1,1,1,0,0,0,1,1};
tipo_juego tu_juego = {0,0,1,1,0,0,0,0};
tu_juego = {1,1,0,0,0,0,0,0};
```

```
for (int i=0; i<=7; i++) {
    mi_juego[i] = 0;
}
```

→ strcpy(cadena)

(* mi_juego[i] = tu_juego[i])

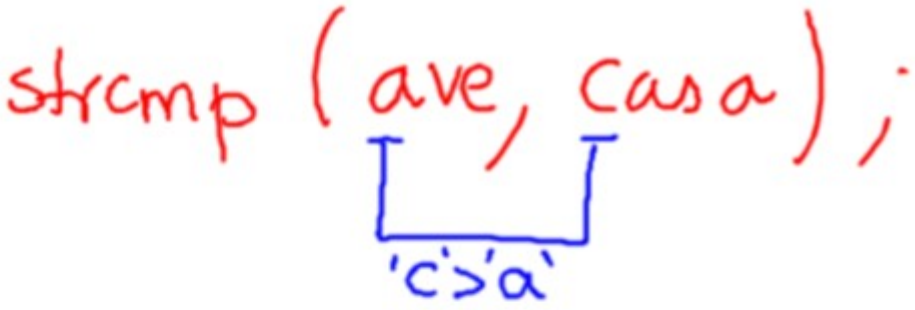
~~mi_juego = tu_juego;~~ ¡PROHIBIDO C++!

```
void MiProc (const tipo_juego mi_juego);
```

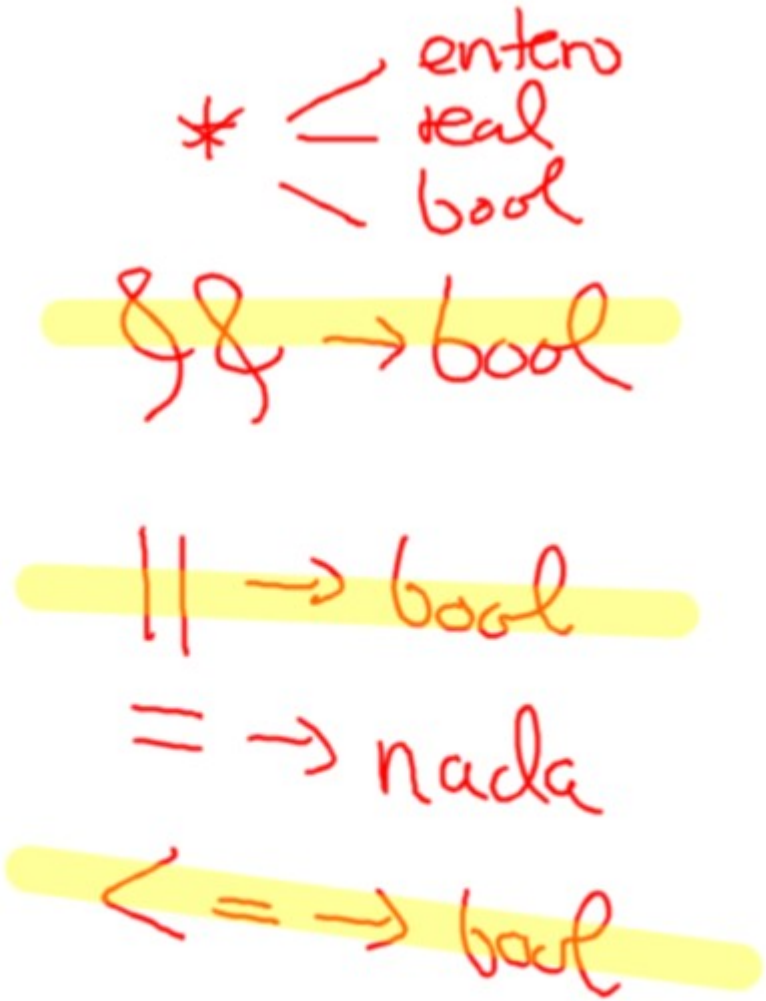
↓ POR VALOR

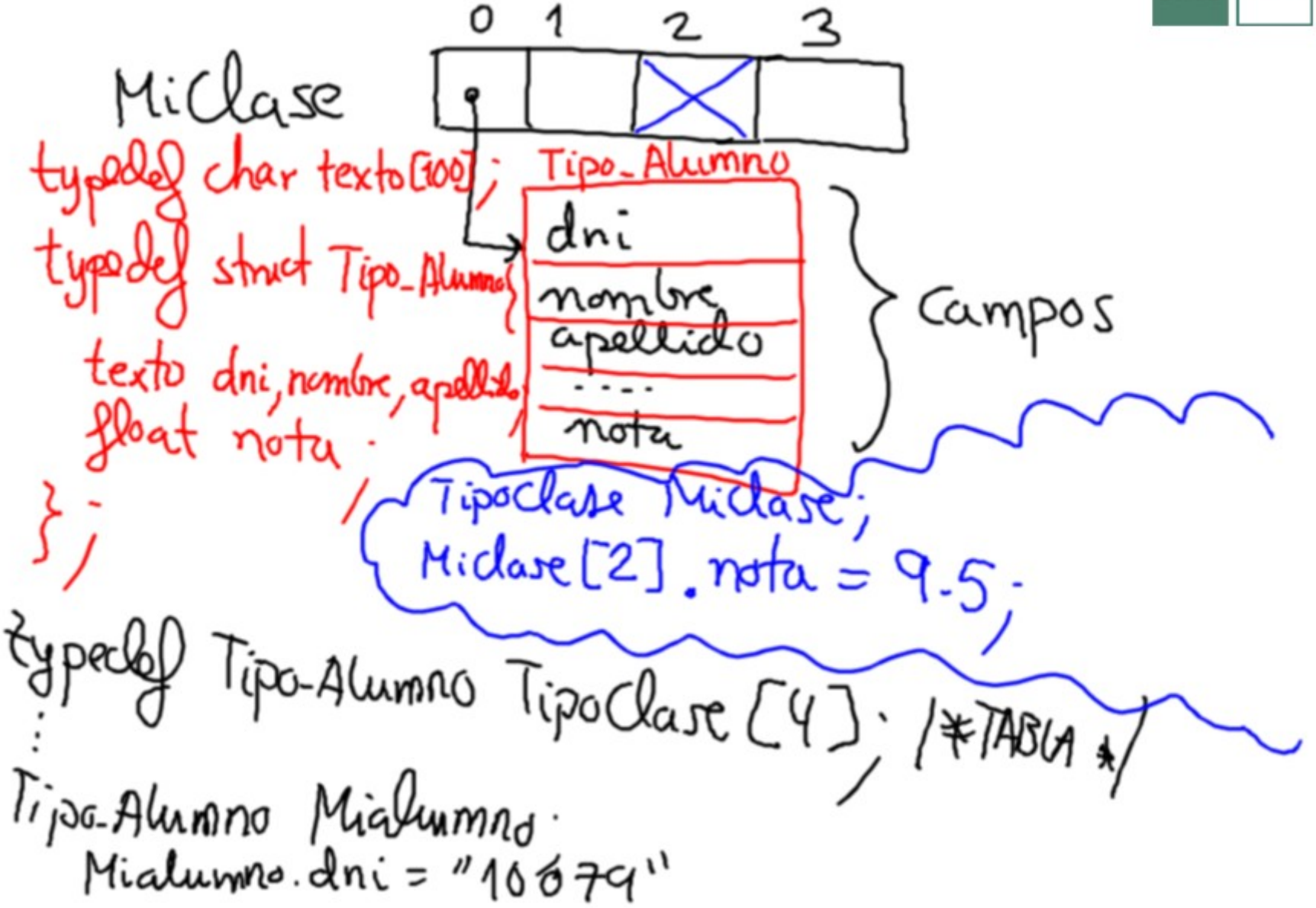
```
Tipo_juego MiFunc (const tipo_juego juego);
```

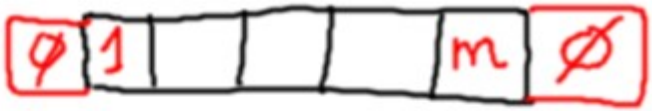
```
typedef char tipo-cadena1 [5];
typedef char tipo-cadena2 [5];

tipo-cadena1 nombre;
tipo-cadena2 apellido;
nombre = apellido;   ¡PROHIBIDO!

apellido = {'G', 'o', 'n', 'z', 'a', 'l', 'e', 'z'};
apellido = "Gonzalez";
```

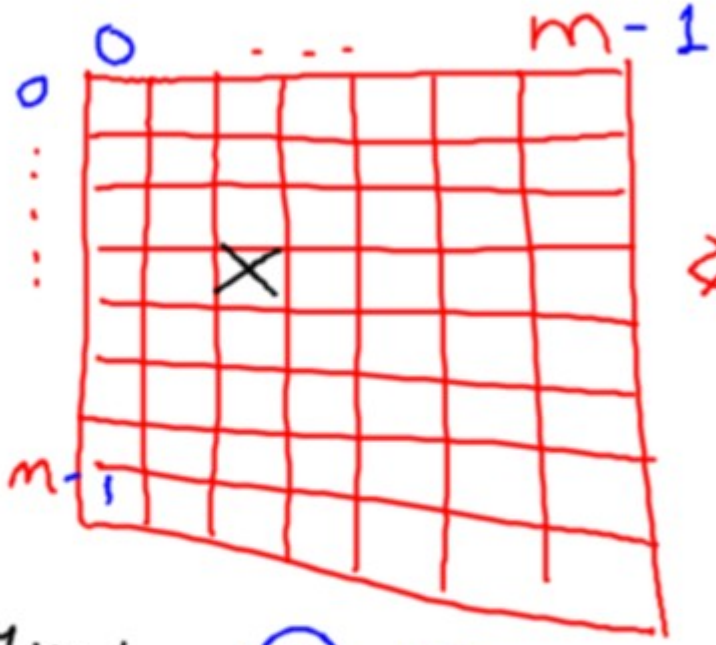


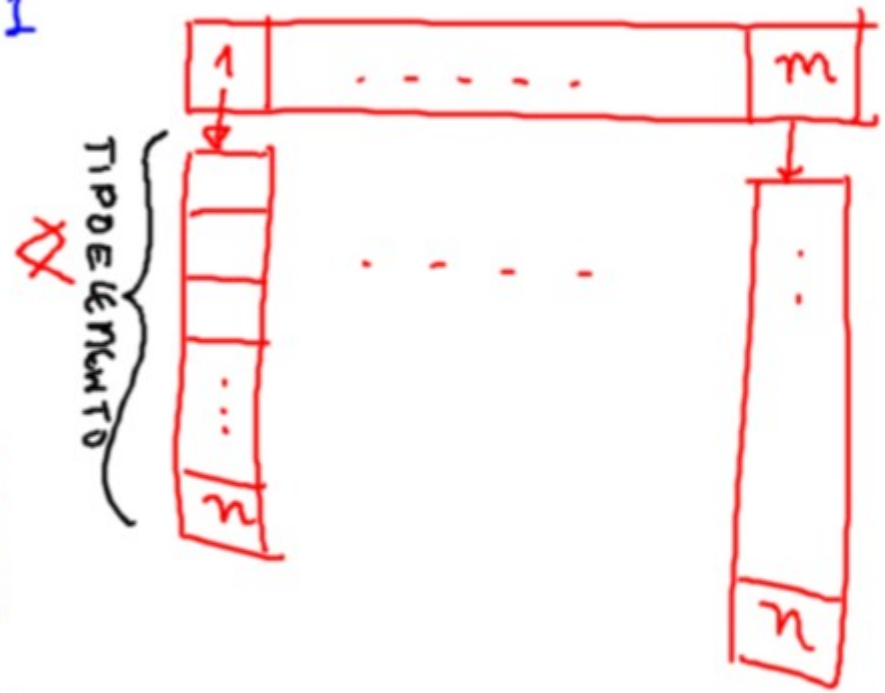

```
int A() {
  int A;
  return A;
}
int main() {
  int A;
  ...
```


*
entero
real
bool
&& → bool
|| → bool
= → nada
<= => bool


0 1 2 3
MiClase
typedef char texto[100];
typedef struct Tipo_Alumno{
texto dni, nombre, apellido;
float nota;
};
Tipo_Alumno
dni
nombre
apellido
....
nota
Campos
TipoClase MiClase;
MiClase[2].nota = 9.5;
typedef Tipo-Alumno TipoClase[4]; /*TABLA*/
Tipo-Alumno Mialumno.
Mialumno.dni = "10679"



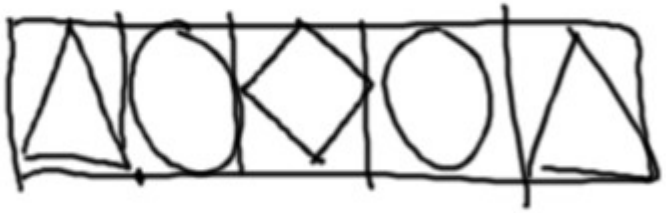




MiMatriz [3] [2] = 27;
FILA COL.

MiMatriz [3] = V; → Asigno un vector V del tipo fila.

```
typedef struct TipoAlumno {
    char nombre[100], apellido[100];
    int nota;
    :
};
```



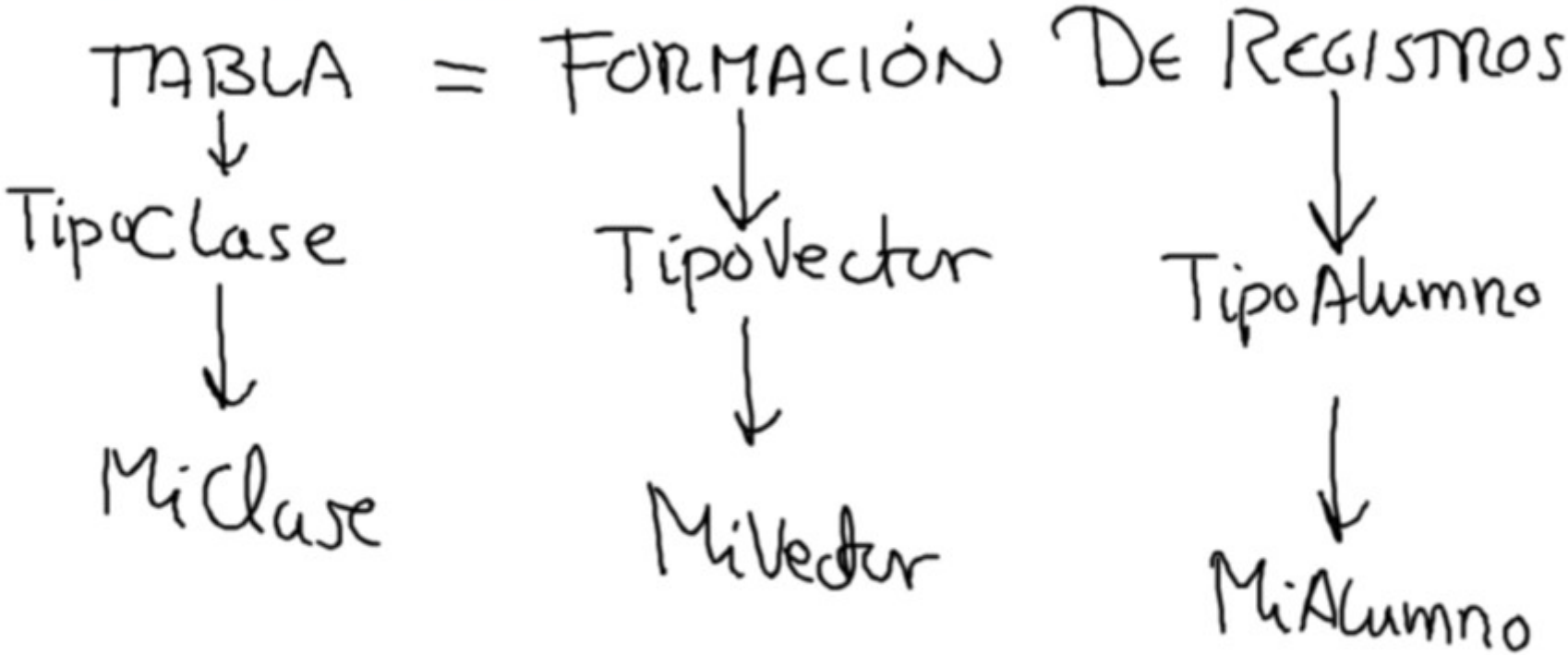
void PintarFiguras(tipoFigura

- Discriminante (enumerado)
- Union (Varios Campos → Sólo Uno).
- struct (Discriminante + Union).

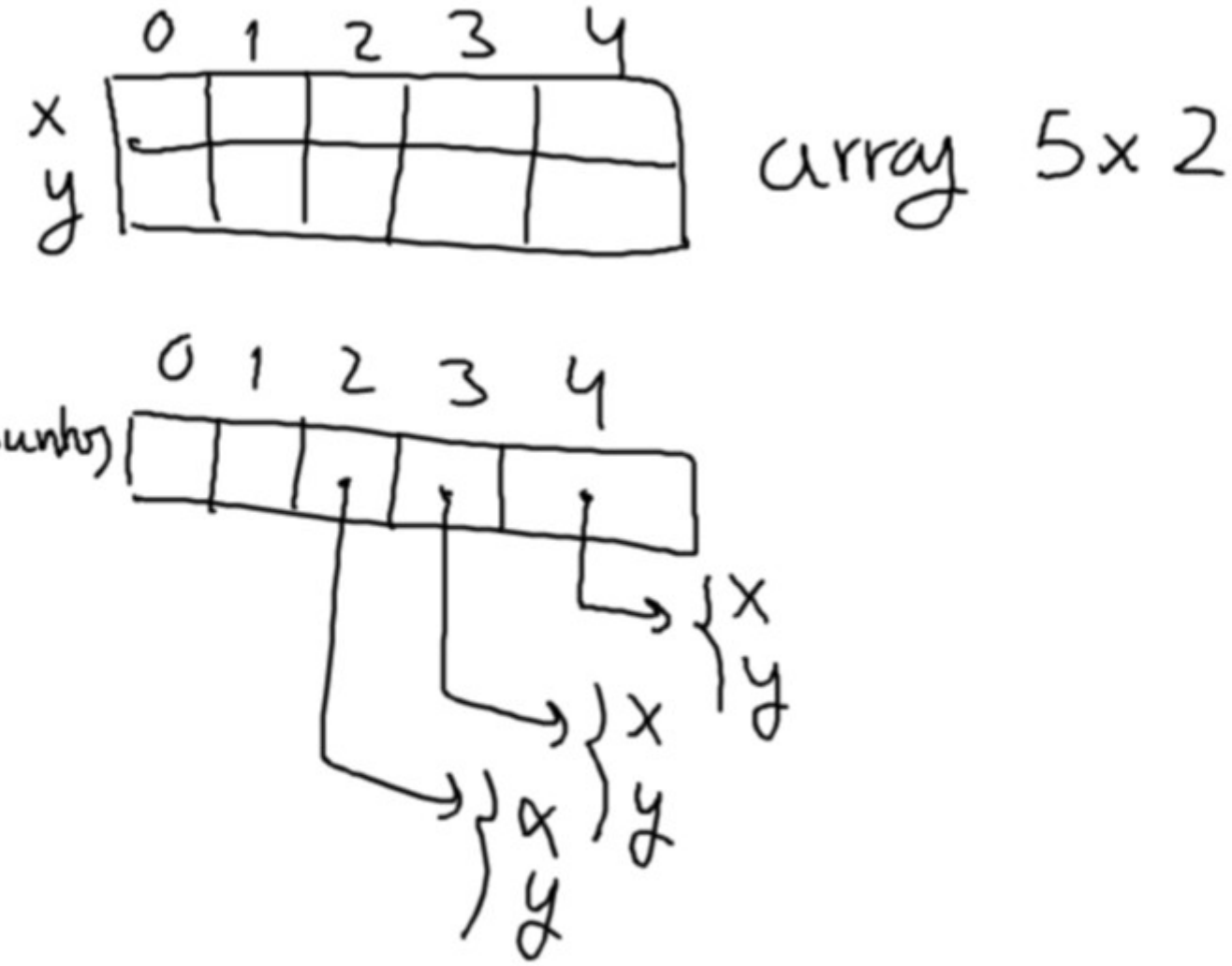
ESQ. ACCIONES	ESQ. DATOS
SECUENCIA	STRUCT (TUPLA)
SELECCION	UNION
REPETICION	FORMACIONES (ARRAY)

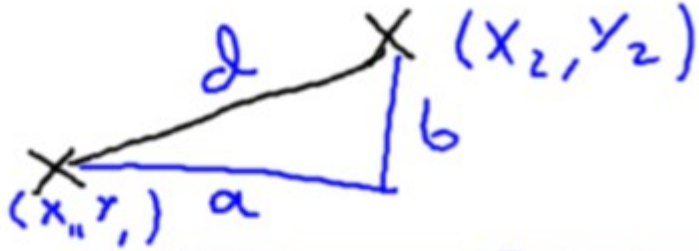
EST. COMBINADAS:

TABLA = FORMACIÓN DE REGISTROS



MiClase[9].nombre


0 1 2 3 4
x
y
array 5x2
0 1 2 3 4
punto
x
y
x
y
x
y



$$d = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}$$

316'84

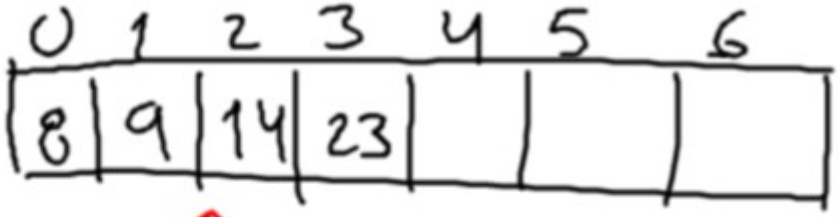
ordenar(fruta)

MisFrutas = TipoFrutas {naranja, pera, limon};

✓ MisFrutas = Ordenar(fruta);

✓ typedef enum TipoFrutas{naranja, pera, limon};

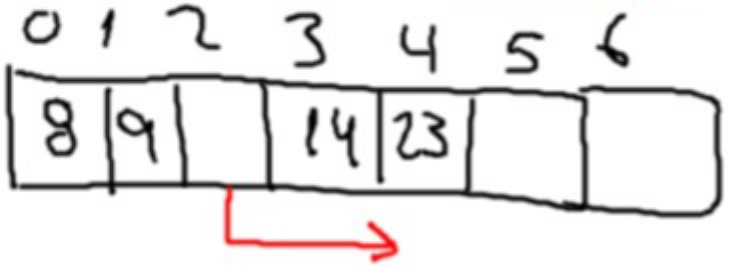
```
for (i=0; i<=n; i++) {
    Dato = dias[i];
};
```



insertar 12

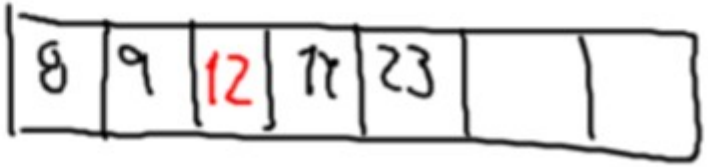
a) Búsqueda / Secuencial
\ Dicotómica

b)



Desplazar derecha

c)

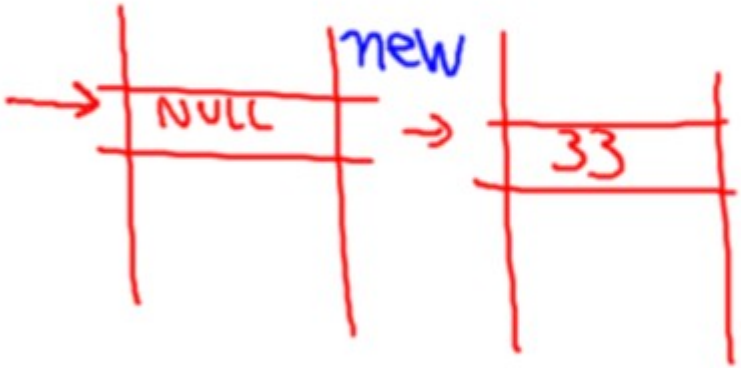


INSERTAR

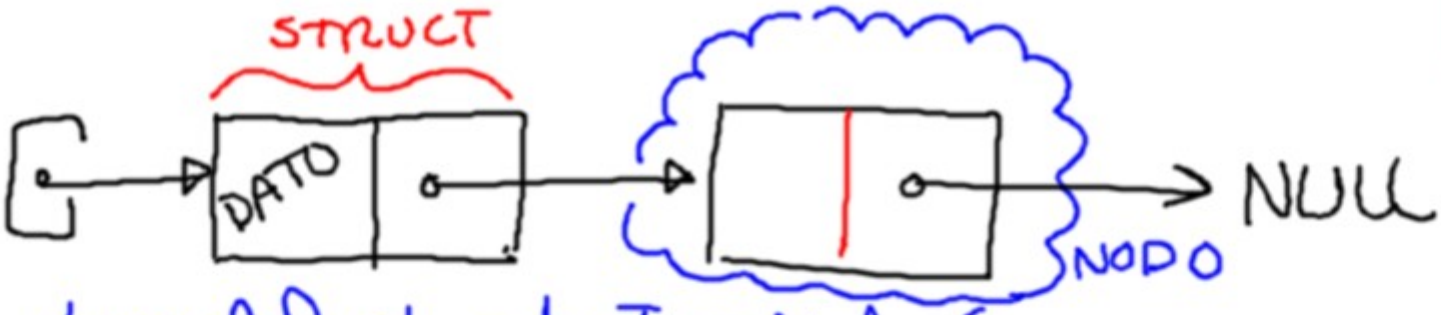
```
if ( elem > V[dcha]) or
   ( elem < V [izda]) then EXIT
```

PUNTEROS : ≡ DINÁMICA


```
typedef int * pt;
pt mi-pt, mi-pt1;
mi-pt= new int;
* mi-pt = 33;
mi-pt1= new int;
mi-pt1 = mi-pt;
```



[mi-pt = 33 ; incompatible]

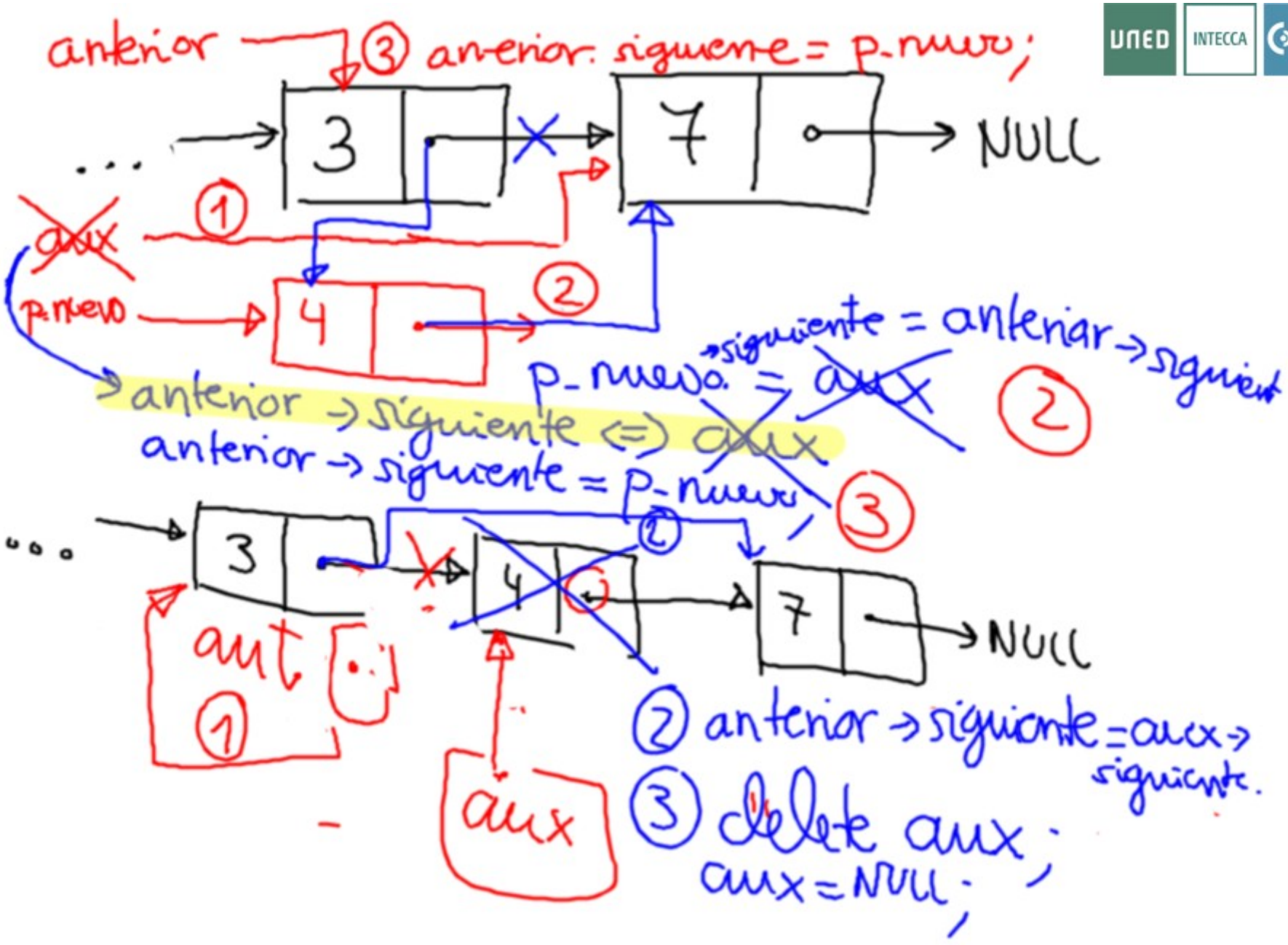


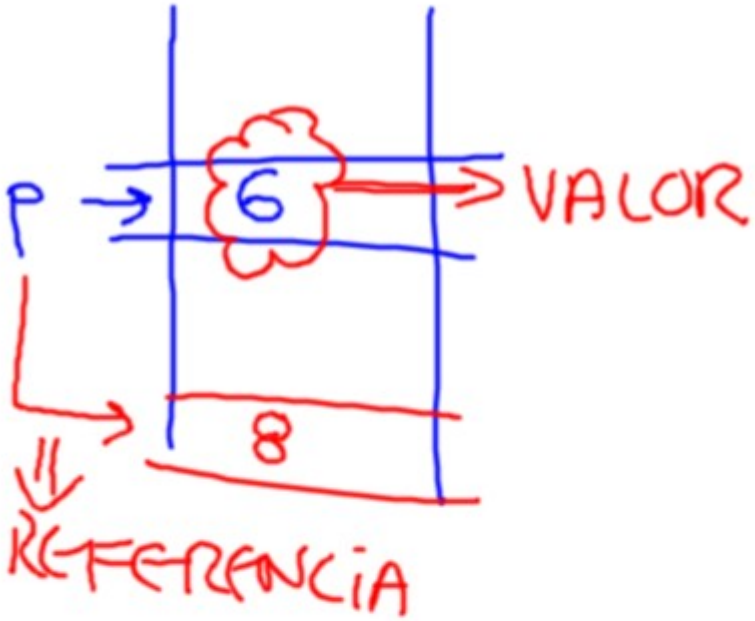

```
typedef struct Tipo_Nodo {
    Tipo_dato primero;
    Tipo_Nodo * siguiente;
};
typedef Tipo_Nodo *Tipolista;
Tipo_lista Mi_Lista;
```

(*mi_lista).siguiente

⇓

mi_lista->siguiente


anterior
3 anterior.siguiente = p_nuevo;
3
7
NULL
...
aux
1
p_nuevo
4
2
anterior -> siguiente <=> aux
p_nuevo.siguiente = anterior -> siguiente
p_nuevo = aux
2
anterior -> siguiente = p_nuevo
3
3
4
7
NULL
aux
1
aux
2 anterior -> siguiente = aux -> siguiente.
3 delete aux;
aux = NULL;

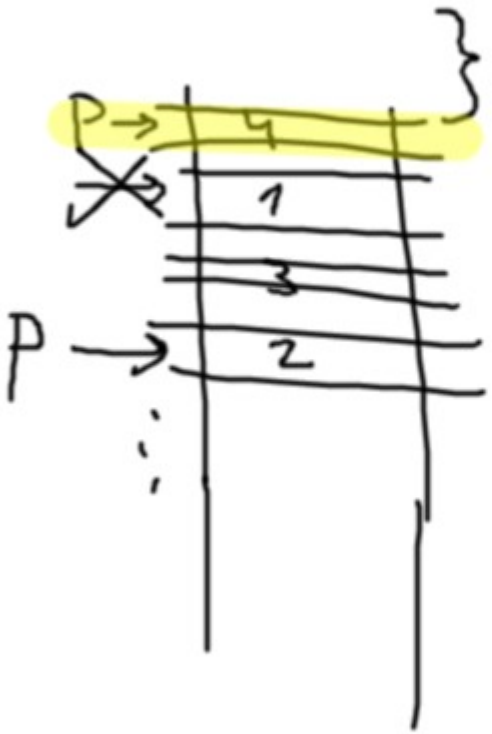

p
6
VALOR
8
REFERENCIA

C.5:

5.2.
```
TipoPunt p1;
TipoReg reg;
p1 = new TipoPunt;
reg.dato = 2;
(*pt1).dato = 3;
```

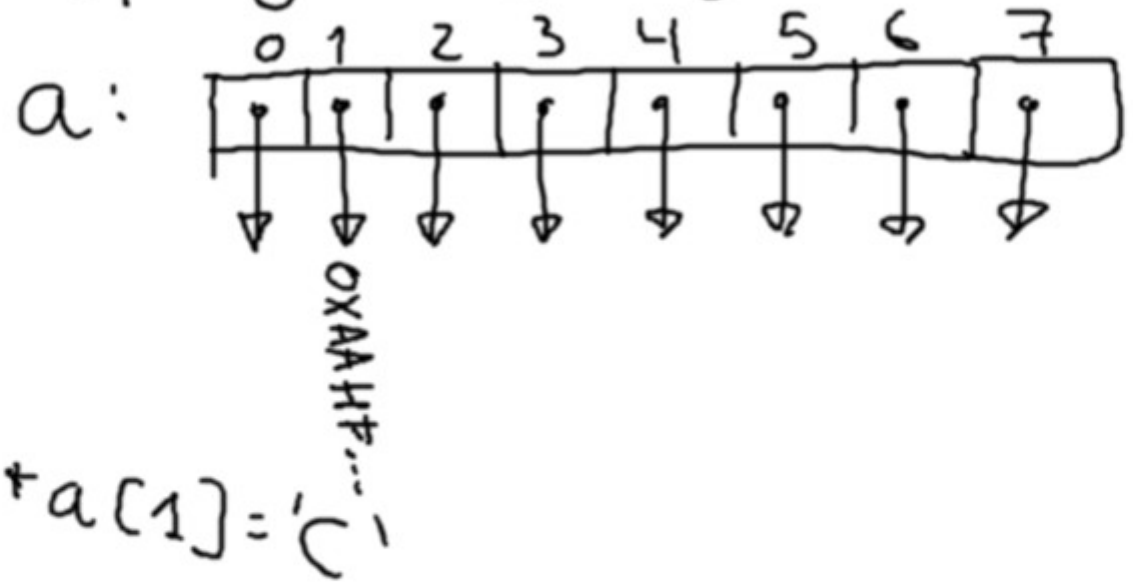
5.3.
```
Tipo_Nodo * secuencia, siguiente;
```

5.5) For (int c1 = 1; c1 <= 4; c1++){
puntero = new TipoPuntero;
*puntero = c1;



5.6.

typedef TCony Conjsubletras[7];



*a[1] = 'C';

5.7. `*p = p->siguiente/23;`

registro

```
typedef struct TipoReg {
        int valor;
        TipoReg *siguiente;
};
typedef TipoReg * TipoPunt;
TipoPunt p;
```