Explicación Parcial 02/07

TEMA 1
CADP 2022

Enunciado Tema 1

La cátedra de CADP necesita un programa para generar el listado de alumnos ingresantes que rendirán el parcial. Para ello, la cátedra **DISPONE** de un listado con todos alumnos que ingresaron este año a la facultad. De cada alumno se conoce su DNI, nombre y apellido, nota obtenida en curso de ingreso (entre 4 y 10), turno (entre 1 y 4) y si estuvo presente o ausente **en cada una** de las 12 clases de práctica.

a) Realizar un módulo que reciba la información de los alumnos y retorne una nueva estructura de datos que contenga sólo aquellos alumnos que podrán rendir el parcial. Para poder rendir el parcial, los alumnos deben contar con al menos 8 asistencias en las 12 clases de práctica.

b) Realizar un módulo que reciba la estructura de datos generada en el inciso anterior, e **IMPRIMA** en pantalla:

- **1)** Apellido y nombre y el DNI de los alumnos que hayan obtenido nota 8 o superior en el ingreso
- **2)** Turno con mayor cantidad de alumnos en condiciones de rendir el examen.
- **3)** Cantidad de alumnos que no posean ningún dígito cero en su DNI.

NOTA: Implementar el programa principal.

Problema general

- *Se dispone de una **lista** de alumnos ingresantes.*
- *El alumno es un **registro** con un campo de tipo vector (para almacenar las asistencias), entre otros. **No es necesario** tener una dimensión lógica para el vector de asistencias ya que está cargado por completo.*

Se pide:

1. *Generar una nueva lista con los alumnos que están en condiciones de rendir el parcial (al menos 8 asistencias). La lista a generar es del mismo tipo que la que se dispone.*

Los siguientes incisos se deben calcular e informar **recorriendo una sóla vez** la lista generada:

2. *Hacer un write de nombre, apellido y dni de los alumnos con nota >= 8. Se compara la nota y si cumple se imprimen los campos del registro.*
3. *Calcular el turno con mayor cantidad de alumnos en condiciones de rendir. Es necesario usar un **vector contador**. Se inicializa a 0 y se incrementa al recorrer la lista. Por último se recorre el vector para obtener el máximo e imprimir el turno.*
4. *Descomponer cada DNI verificando que no tiene el dígito 0. Si es así incrementamos una variable. Este punto se calcula al recorrer la lista y se imprime una única vez al final.*

Solución Tema 1

```
program ingresantes;
const
  cant_clases = 12;
  cant_turnos = 4;
type
  rango_notas = 4..10;
  rango_turnos = 1..cant_turnos;
  rango_clases = 1..cant_clases;
  vector_clases = array[rango_clases] of boolean;
  alumno = record
       DNI : integer;
       nombre_apellido : string;
       nota : rango_notas;
       turno : rango_turnos;
       asistencias : vector_clases;
      end;

  lista = ^nodo;
      nodo = record
      dato : alumno;
      sig : lista;
  end;

  vector_turnos = array[rango_turnos] of integer;
```

```
//PROGRAMA PRINCIPAL
var
  L1,L2 : lista;
begin
     cargarAlumnos(L1); //SE DISPONE
     procesarAlumnos(L1,L2); //PUNTO A
     procesar_e_imprimir(L2); //PUNTO B
 end.
```

Solución Tema 1 - Inciso a

```
procedure agregarAdelante(var L : lista; a : alumno);
var
   aux : lista;
begin
   new(aux);
   aux^.dato := a;
   aux^.sig := L;
   L := aux;
end;

procedure procesarAlumnos(lista_alumnos : lista; var lista_rinden : lista);
var
   aux : alumno;
begin
   lista_rinden := nil;
   while (lista_alumnos <> nil) do
   begin
      aux := lista_alumnos^.dato;
      if (cumple_condicion(aux.asistencias)) then
        agregarAdelante(lista_rinden,aux);
      lista_alumnos := lista_alumnos^.sig;
   end;
end;
```

Solución Tema 1 - Inciso a

```
// opción 1: usamos un for
function cumple_condicion(asistencias : vector_clases) : boolean;
var
   i, cant : integer;
begin
   cant := 0;
   for i:=1 to cant_clases do
      if (asistencias[i]) then
        cant := cant + 1;
   cumple_condicion := (cant >= 8);
end;
```

```
// opción 2: usamos un while. ¡Más eficiente!
function cumple_condicion(asistencias : vector_clases) : boolean;
var
   i, cant : integer;
begin
   cant := 0;
   i := 1;
   while (i <= 12) and (cant < 8) do
   begin
      if (asistencias[i]) then
        cant := cant + 1;
      i := i+1;
   end;
   cumple_condicion := (cant = 8); // if cant = 8 then cumle_condicion := true else cumple_condicion := false;
end;
```

Solución Tema 1 - Inciso b

```
procedure procesar_e_imprimir(L : lista);
var
  turnos : vector_turnos;
  cantAlumnos : integer;
begin
  inicializar(turnos);
  cantAlumnos := 0;
  while (L <> NIL) do
  begin
     //Inciso 1
     if (L^.dato.nota >= 8) then
       writeln(L^.dato.nombre_apellido, l^.dato.DNI);
     //Inciso 2
     turnos[L^.dato.turno] := turnos[L^.dato.turno] + 1;
     //Inciso 3
     if (cant_digitos(L^.dato.dni,0) = 0) then
       cantAlumnos := cantAlumnos + 1;

     L := L^sig;
  end;
  //inciso 2
  writeln(maximo(turnos));
  //inciso 3
  writeln(cantAlumnos);
end;
```

```
procedure inicializar(var v : vector_turnos);
var
  i :  rango_turnos;
begin
  for i:= 1 to cant_turnos do
     v[i] := 0;
end;
```

Solución Tema 1 - Inciso b2

```
function maximo(v : vector_turnos) : rango_turnos;
var
  i, max_turno : rango_turnos;
  maxAlumnos : integer;
begin
  maxAlumnos := -1;
  for i:= 1 to cant_turnos do
     if (v[i] > maxAlumnos) then
     begin
       maxAlumnos := v[i];
       max_turno := i;
     end;
  maximo := max_turno;
end;
```

Solución Tema 1 - Inciso b3

```
// Otra opción: no envío el parámetro digito_buscado, y esta función sólo cuenta la cantidad de ceros

function cant_digitos(num,digito_buscado : integer) :integer;
var
   dig,cant : integer;
begin
   cant := 0;
   while (num <> 0) do
   begin
      dig := num MOD 10;
      if (dig = digito_buscado) then
        cant := cant + 1;
      num := num DIV 10;
   end;
   cant_digitos := cant;
end;
```

```
// Otra opción: retorno TRUE si no tiene ceros, o FALSE si aparece algun cero. ESTA OPCION PUEDE SER MAS EFICIENTE!

function tiene_ceros(num : integer) : boolean
var
   encontre_un_cero : boolean;
   dig : integer;
begin
   encontre_un_cero := false;
   while (num <> 0) and (not encontre_un_cero) do
   begin
      dig := num MOD 10;
      if (dig = 0) then
        encontre_un_cero := true
      else
        num := num DIV 10;
   end;
   tiene_ceros := encontre_un_cero;
end;
```