FINAL CADP – 09/08/2022

Apellido y Nombre: ..... DNI: .....

1. **Práctica:** Se dispone de la información de los participantes inscriptos a una carrera (a lo sumo 5000). De cad participante se tiene DNI, nombre y apellido, categoría (1..5) y fecha de inscripción. Se pide implementar u programa que guarde en una estructura adecuada los participantes de aquellas categorías que posean a lo sumo 5 inscriptos. Se sabe que cada participante se puede anotar en una sola categoría.

2. Indique para las siguientes proposiciones, si son **Verdaderas** o **Falsas**. **Justifique** cada caso.
   - No es posible la utilización de las variables globales para la comunicación entre los módulos de un programa.
   - Siempre es posible realizar la eliminación de un elemento en un vector.
   - Un programa modularizado puede no ser correcto.
   - El acceso a un elemento de una estructura de datos lineal sólo es posible a través de un recorrido secuencial.

3. Dada la siguiente declaración de tipos de datos y variables, justificar para cada sentencia numeradas son válidas o inválidas:

```
program ejercicio_3;
type
  cadena50 = string[50];
  cliente = record
    DNI: cadena50; ape_nom: cadena50;
  end;
  clientes = ^nodo;
  nodo = record
    dato: cliente; sig: clientes;
  end;
var
  c: cliente; cli: clientes; cli_esp: clientes;
```

```
begin
   1. read(c);
   2. new(c);
   3. cli := nil;
   4. new(cli);
   5. cli_esp := cli;
   6. dispose(cli);
   7. read(cli_esp^.DNI);
   8. write(cli_esp^.DNI);
end.
```

4. Describa las formas de comunicación entre módulos vistas en la materia.

5. Teniendo en cuenta las referencias, calcule e indique la cantidad de **memoria estática, memoria dinámica** y el **tiempo de ejecución**. Muestre cómo se obtienen los resultados.

```
program ejercicio_5;
type
 cadena30 = string[30];
 categorias = 1..5;
 participante = record
   ape_nom: cadena30;
   categ: categorias;
   tiempo: real;
 end;
 vector = array [1..20] of ^participante;
var
 p: vector; i:integer; c: categorias;
 ayn: cadena30;
begin
   for i:= 1 to 10 do begin
     new(p[i]); read(c); read(ayn);
     p[i]^.categ:= c; p[i]^.ape_nom:= ayn;
     p[i]^.tiempo:=0;
   end;
   for i:= 10 downto 5 do
     dispose(p[i]);
end.
```

Referencia	
Char	1 byte
Integer	4 bytes
Real	8 bytes
Boolean	1 byte
String	Longitud + 1
Puntero	4 bytes

```pascal
type
 participante = record
  dni :integer;
  nombre : string [10];
  apellido : string [10];
  categoria : integer;
  fecha : integer;
  end;

 vectorParticipantes = array [1..5000] of participante;

 vectorContador = array [1..5] of integer;

 vectorNuevo = array [0..250] of participante; //como maximo van a haber 250 participantes guardados

//procedure cargarVectorPartipantes (v:vectorParticipantes;dimL1:integer); //se dispone

procedure iniciarlizarVectorContador(var v:vectorContador);
 var
 i:integer;
 begin
  for i:= 1 to 5 do
   begin
    v[i]:=0;
   end;
 end;

procedure cargarVectorContador(var v:vectorContador;p:vectorParticipantes;dimL:integer);
  var
   i:integer;
  begin
   for i:= 1 to dimL do
    begin
     v[p[i].categoria]:=v[p[i].categoria] + 1;
    end;

  end;

procedure recorrerVector (v:vectorParticipantes;dimL1:integer;vecContador:vectorContador;var nuevoVector:vectorNuevo;var dimL2:integer);
 var
 i:integer;
 begin
  for i:=1 to dimL1 do
   begin
    if (vecContador[v[i].categoria]<=50) then
     begin
      dimL2:=dimL2+1;
      nuevoVector[dimL2]:=v[i];
     end;
   end;
 end;
var
v1:vectorParticipantes;
v2:vectorNuevo;
dimL1,dimL2:integer;
v:vecContador;
begin
dimL1:=0;
dimL2:=0;
//cargarVectorPartipantes(v1,diml1) (se dispone);
iniciarlizarVectorContador(v);
cargarVectorContador(v,v1,dimL1);
recorrerVector(v1,dimL1,v,v2,dimL2);
end.
```

2.

a) Falso, es posible ya que las variables globales pueden ser utilizadas a lo largo de todo el programa. Sin embargo esto trae consigo problemas, por lo cual no es una practica recomendable.

b) Falso, podría ocurrir que el elemento no exista o no se encuentre en una posición valida.

c) Verdadero, un programa puede estar modularizado pero no resolver los problemas para los cuales se diseño, y por lo tanto no ser correcto.

d) Falso, un arreglo es una estructura de datos lineal pero también es indexada. Asi que es posible acceder a un elemento N usando su índice correspondiente sin hacer un recorrido secuencial.

3.

1. Invalida, no es posible leer directamente un registro desde teclado. Debo leer cada uno de sus campos.

2. Invalia. Cliente es un registro, no un puntero al que le tenga que reservar memoria dinámica.

3. Valida, clientes es un puntero a nodo, por lo cual es valido inicializarlo dándole el valor nil;

4. Valida, new esta reservando memoria dinámica para cli, que es un puntero a nodo.

5. Valida, ahora cli_esp y cli apuntan a la misma dirección de memoria.

6. Valida, libero la memoria que reserve en la línea 4 para que pueda volver a ser utilizada por el programa.

7. invalida, no puedo acceder directamente al dato DNI del nodo al que apunte cli_esp. La línea correcta para hacerlo seria: cli_esp^.dato.dni

8. invalida, no puedo acceder directamente al dato DNI del nodo al que apunte cli_esp. La línea correcta para hacerlo seria: cli_esp^.dato.dni

4.

Por parámetros:

Los Parámetros son variables que tienen como característica transferir información entre módulos

Por variables globales:

Es desaconsejable, pero los modulos podrían comunicarse entre si mediante variables globales que son accesibles en todo el programa.

5.

Memoria estatica:

Vector = 20*4 = 80 bytes

I:integer = 4 bytes

C:categorías = 4 bytes

Ayn:cadena30 =31bytes

Total = 80+4+4+31 = 119 bytes

Memeria dinámica:

Participante=record

Ape_nom:cadena30 = 31bytes

Categ:categorías = 4bytes

Tiempo:real = 8bytes

End; total =31+4+8 = 43 bytes

El for hace 10 new(p[i]), es decir 10*43bytes = 430 bytes

For :=10 down to 5 son 6 iteraciones del dispose, es decir 6*43 bytes = 258 bytes

Total memoria dinámica = 430-258 = 172 bytes

Tiempo de ejecución:

1) for i:=1 to 10 , es igual a 3*10+2 = 32 ut

2) cuerpo = 3 ut, multiplicado por 10 iteraciones es igual a 3*10 = 30ut

3) for i:10 downto 5, es igual a 3*6+2 = 20 ut

Total tiempo de ejecución = 32+30+29 = 82 ut