**FINAL CADP – 05/07/2022**

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

1. **Práctica:** Se dispone de la información de los productos de un supermercado. De cada producto se tiene Código, Nombre, Rubro (1..20) y precio. Se pide implementar un programa que guarde en una estructura adecuada los productos de los rubros que tengan 10 productos.

2. Indique para las siguientes proposiciones, si son **Verdaderas** o **Falsas**. **Justifique** cada caso.
   - En la técnica de corrección de debugging es necesario analizar los casos límites del problema.
   - Un vector siempre se utiliza teniendo en cuenta la dimensión lógica.
   - Una función puede devolver un tipo de dato registro, real, booleano, integer, entre otros.
   - Un programa que utiliza sólo variables globales no requiere modularización.

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

```
program ejercicio3;
type
  cadena100 = string[100];
  cliente = record
    codigo - integer; tel: integer; dir: cadena100;
  end;
  clientes = ^nodo;
  nodo = record
    datos: cliente; sig: clientes;
  end;
var
  c: cliente; cli: clientes;
```

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

4. Describa las características de una estructura del tipo de dato vector y describa los pasos necesarios de la operación de búsqueda de un elemento en dicha estructura.

5. Teniendo en cuenta las referencias, calcule e indique la cantidad de **memoria estática** y el **tiempo de ejecución**. Muestre cómo obtiene resultado.

```
program ejercicio5;
type
 cadena20 = string[20]; notas = 2..10;
 alumno= record
            ape_nom: cadena20;
            nota: integer;
          end;
 vector = array [1..10] of ^alumno;
var
 v: vector; i:integer; sum: integer; nota: notas;
 apeNom: cadena20;
begin
 for i:= 1 to 10 do begin
   new(v[i]); read(nota); read (apeNom);
   v[i]^.nota:= nota; v[i]^.ape_nom:= apeNom;
 end;
 sum := 0;
 while (sum < 200) do begin
   read (nota); sum := sum + nota;
 end;
end.
```

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

1.

```
cadena = string[10];
producto = record
 codigo:integer;
 nombre:cadena;
 rubro:integer;
end;

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

vectorContador = array [1..20] of integer;

vectorProductos = array [1..200] of producto; //puede haber como maximo 200 productos ya que son 20 rubros y para guardarse aqui tienen que tener 10 productos.

procedure iniciarVectorContador (var v:vectorContador);
var
i:integer;
begin
for i:=1 to 20 do
 begin
  v[i]:=0;
 end;
end;


procedure contar (l:lista;var v:vectorContador)
begin
 while (l<>nil)do
  begin
   v[l^.dato.rubro]:= v[l^.dato.rubro]+1;
   l:=l^.sig;
  end;
end;

procedure recorrerLista (l:lista; var v:vectorProductos;var diml:integer; c:vectorContador);

begin
 while (l<>nil) do
  begin
   if (v[l^.dato.rubro] = 10) then
    begin
     v[diml] := l^.dato;
     diml:=diml+1;
    end;
   l:=l^.sig;
  end;
end;

var
l:lista;
v:vectorProductos;
vc:vectorContador;
diml:integer;

begin
  l:=nil;
  diml:=0;
  cargarLista(l:lista); //se dispone
  contar (l,vc);
  recorrerLista(l,v,diml,vc);
end.
```

2.

a) Falso, no es necesario. Analizar los casos borde es mas bien propio de la técnica de testing.

b) Falso, no necesariamente. Por ejemplo un vector contador [1..5] para llevar cuantos inscriptos tienen 5 categorias de un concurso no necesita una dimensión lógica, simplemente inicializamos el vector poniendo cada valor en 0.

c) Falso. Una función no puede devolver un tipo de dato registro. Solo puede devolver tipos de datos simples, registro es un tipo de dato definido por el usuario.

d) Falso. Un programa requiere modularizacion si esta es beneficiosa para descomponer los problemas que quiere resolver, reutilizar código, facilitar el desarrollo,etc. Independientemente de que use variables globales o no.

3.

1. Invalida, no se puede leer desde teclado directamente un registro.

2. Invalida, new se usa para reservar memoria dinámica para la variable a la que apunta un puntero. C es una variable de tipo registro, no un puntero.

3. Invalida, clientes es un puntero a nodo y guarda una dirección de memoria. No es posible asignarle una dirección de memoria por lectura como se pretende hacer con read.

4. Invalida, nil es una palabra reservada que representa un valor especial que puede tomar un puntero que no apunta a ningún lugar en particular. C es una variable de tipo registro.

5. Valida, cli es una variable de tipo puntero cli:=nil; le asigna este valor especial que indica que el puntero no apunta ninguna dirección en particular.

6. valida, dispose libera la memoria dinámica alocada a un puntero, cli es de tipo puntero. Sin embargo vale destacar que en este caso nunca se asigno memoria dinámica a cli mediante new

7. invalida, se esta queriendo acceder a la variable código del dato del nodo al que apunta cli, pero no se esta haciendo correctamente. Tambien hay que acceder a dato, no se puede accedera  código directamente. La línea correcta seria: cli^.dato.codigo

8. valida, se esta accediendo correctamente a la variable código del registro c. si esta tiene valor se imprimirá en pantalla.

4.

Vector: Tipo de arreglo con un solo índice.
Caracteristicas: Homogenea, estatica, Indexada. Tipo de dato definido por el programador.

Busqueda de un elemento:

Busqueda líneal: Comienzo al principio de la estructura, comparo el elemento con el que estoy buscando, si es igual termina la búsqueda, sino avanzo al siguiente elemento. Repito hasta que encuentre el elemento, recorra toda la dimensión lógica del arreglo o si el vector esta ordenado hasta que el elemento que analizo sea mayor (menor) que el elemento que estoy buscando.

Busqueda dicotómica (vector ordenado):

Se evalua el elemento que esta en la posición del medio. Si es el elemento que estoy buscando termina la búsqueda. Si no es el elemento que estoy buscando comparo si es mayor o menor que el elemento del medio. Elijo la mitad (izquierda o derecha) que me convenga según si es mayor o menor. Repito hasta encontrar el elemento o agotar la búsqueda.

5.

Memoria estatica:

V:vector = 10*4 =40

I,sum:integer = 4*2=8

Nota:notas = 4

Ape_nom:cadena20 = 21

Total = 40+8+4+21=73bytes

Tiempo de ejecución:

For i:=1 to 10  = 3*10+2 = 32

Cuerpo = 2 *10 = 20

Asignación = 1

While (sum<200) = 1*n+1 = 1*100+1

Cuerpo = 2 *100 = 200

Total = 32+1+20+101+200 = 354 ut