FINAL CADP – 05/07/2022

Apellido y Nombre: ...SAMBIDO ADRIAN..... **DNI:** ...18251258.....

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

t(1) = 3.10+2 = 32 UT

t(2) = ASIGNACIONES 2UT
10 * 2UT = 20 UT

t3 = ASIGNACIÓN 1 UT

t(4) (N+1)

t(5) = ASIGNACIÓN + SUMA
2UT => N.2

t(1) + t(2) = 32 + 20 = 52

t3 + t4 + t5 = 1 + (N+1) + N2 = 3N+2

t(10) = 52 + 3N+2 = 3N+54

O(10) = N

```
while (L <> nil) DO
    IF (V [L^.dato.cubo] = 10)
        then agregarAdista (LN, L^.dato)
```

SAMBIDO ADRIAN 4091/1

2) ITEM

2.1) DEBUGGING: En la técnica de depuración es necesario analizar los casos límites para una correcta evaluación del algoritmo

2.2) FALSO, DEPENDIENDO DE CUAL SEA LA TAREA SE PUEDE TRABAJAR O NO CON LA DIMENSIÓN LÓGICA, Y ¿QUÉ EN OTROS CASOS ES NECESARIO RECORRER TODO EL VECTOR Y AQUÍ SE PUEDE TRABAJAR CON LA DIMENSIÓN FÍSICA.

3.2) Una función sólo puede devolver dato simple, no puede devolver un tipo de dato registro.

2.4) La utilización de variables globales no conviene la no utilización de módulos, ya que la utilización de módulos es para tener mayor legibilidad y direccionamiento del programa.

---

3) ITEM

1) - Sentencia inválida ya que no se puede hacer un READ de un registro lo correcto sería hacerlo sobre un campo del mismo Ej. C. Codigo.

2) - Sentencia correcta se está reservando en memoria dinámica lugar para una variable de tipo registro.

3) - Sentencia inválida se está tratando de leer un tipo de dato puntero aunque el lenguaje lo permita lo correcto sería leer un campo apuntado por el puntero. (Del siguiente).

4) Sentencia inválida se le está asignando nil a un tipo de dato registro.

5) - Sentencia válida ya que se está asignando nil a un puntero.

6) Sentencia válida, se está liberando la memoria del puntero.

7) Sentencia inválida, la correcta sería READ(CLI^.DATO.CODIGO)

8) Sentencia válida es la manera correcta de acceder al campo de los registros.

---

4) ITEM

4.1 Un tipo de dato vector es una estructura lineal, indexada, estática, homogénea.

4.2 Para hallar un elemento en dicha estructura, puede hacerse de manera, secuencial, directa o a través de una búsqueda dicotómica

Pasos
- CONFIRMAR QUE TENGO ELEMENTOS.
- UTILIZA ALGUNO DE LOS ACCESOS ANTES MENCIONADOS.
- EN EL CASO DE LA BUSQUEDA SECUENCIAL RECORRER HASTA ENCONTRAR EL ELEMENTO O HASTA QUE SE LO RECORRIO TODO. (O ACCEDER DIRECTAMENTE A TRAVEZ DE UN ÍNDICE)

ACCEDER AL ELEMENTO.
PROCESARLO.

---

5) ITEM

MEMORIA ESTÁTICA

EN LAS DECLARACIONES SE PIDE MEMORIA PARA.

V: Vector. Quién ocupa 10 posiciones de 4 bytes = 40 Bytes + [(21+4).10] = 40 + 250 = 290 Bytes

SUM = 4 bytes

NOTA = 9 x 4 = 36 bytes

DPENOM: 21 bytes

290 + 4 + 36 + 21 = 351 Bytes

TIEMPO DE Ejecución

```
Procedure LeerDatosyProcesar (VAR Vec : Vector ; Lis : Lista);
VAR
Begin
  While (Lis <> Nil) do Begin
    Vec[Lis^.dato.Rubro] := Vec[Lis^.dato.Rubro] + 1;
    Lis := Lis^.Sig;
end.
```

```
                                    Vec: Vector.
Procedure GuardaRegistros (VAR Vec2: Vector2 ; Lis : Lista);
  Var I: Integer; Aux: Lista
    Begin
      For I := 1 to 20 do          Begin ← Aux := Lis;
        IF (Vec[I] = 10) then      Begin
          While (Aux <> Nil) and (Vec[I] <> Aux^.dato.Rubro) do
            Begin
              Aux := Aux^.Sig;
            end.
          IF (Lis <> Nil) then
    end end   Vec2[i] := Lis^.dato;
VAR Lis: Lista
Vec: Vector; Vec2: Vector2;
  Begin
    InicializarVector(Vec) (Primera);
    LeerDatosyProcesar(Vec, Lis);
    GuardaRegistros(Vec,Vec2; Lis);
end.
```

SAMBEDO ADRIAN Nº 4091/1

PROGRAMA SAMBEDO FINAL CADP 2022 — 5/07/2022

```
type  Vector2 = Array [1..20] of Producto (Guarda los productos con 10 articulos)
      Lista = ^Nodo;   Lista (con la que se cuenta).
      Vector = Array [1..20] of Integer (Vector Contador)
      PRODUCTO = Record
                   CÓDIGO : INTEGER (REGISTRO DE PRODUCTOS
                   NOMBRE : STRING;
                   RUBRO : 1..20;
                   PRECIO : REAL;
                 end.
      NODO = Record
               DATO : Producto;   (Registro de la Lista)
               SIG : Lista;
             end.
```

```
Begin
  Write ('Ingrese Codigo'); Read (Reg.Codigo);
  Write ('NOMBRE'); Read (reg.nombre);
  Write ('Rubro'); Read (Reg.Rubro);
  Write ('Precio'); Read (reg.Precio);
End.
```

```
Procedure Inicializa Vector (VAR Vec : Vector);
 VAR I : Integer.
    Begin
      For I := 1 to 20 do Begin
        Vec[I] := 0;
      end.
    end.
```