

4. Calcule el tiempo de ejecución del programa del punto 3. Mostrar los valores intermedios para llegar al resultado y justificar.

5. Indique Verdadero o Falso. Justifique en todos los casos:

a. Incluir módulos dentro de un programa implica que el programa es más eficiente que otro programa que realiza la misma tarea pero sin utilizar módulos.

b. El siguiente programa es válido.

```
program ejercicio;
function auxiliar(val:integer): integer;
begin
    val:= val * val;
    auxiliar:= val;
end;
procedure calculo(c: Integer; var b:integer);
begin
    b:= b + c DIV 4;
end;
var
    a,b:integer;
begin
    a:= 16;
    b:= 6;
    calculo(auxiliar(a),b);
end;
```

c. No siempre es posible declarar un tipo subrango donde su tipo base sea cualquiera de los tipos simples en la teoría.

d. Un programa que utiliza un repeat until puede reescribirse utilizando un while.

e. La comunicación entre el programa y los módulos no sólo se puede hacer utilizando parámetros.

2) Ambos programas son válidos. El programa A primero declara un Array en el apartado type (variables definidas por el usuario), para luego referenciarlo con una variable "d" en el programa principal. De la misma manera, el programa B declara aquel Array directamente en el apartado var del P. Principal, siendo algo válido en Pascal. En el programa principal, el funcionamiento es el mismo. Pero si queremos modularizar, el programa B no facilita la reutilización de ese arreglo, ya que se encuentra declarado debajo solamente para el P. Principal. En el caso de que el apartado var se encuentre por encima de los módulos, estos podrán usar el arreglo, pero el P. Principal se verá obligado a declarar todas sus variables allí arriba de manera global, siendo algo muy peligroso para la seguridad del programa.

4) Línea 2: 1UT

Línea 3:

N=100; C=3; Cuerpo=4

--> $3(100+1) + 100(4) = 703UT$

Línea 10:

N=100; Cuerpo=2

--> $(3*100 + 2) + 100(2) = 502UT$

TOTAL: $1 + 703 + 502 = 1206UT$

5)

a. Verdadero. Modularizar no implica eficiencia respecto a tiempo de ejecución, pero sí a memoria. Cuando se modulariza y se declaran variables internas en los módulos, se procura que las variables locales que se alojen en la memoria dinámica solamente duren durante la ejecución del módulo. En otras palabras, aunque no sea demasiado más eficiente, si lo es si se modulariza y se asignan adecuadamente las variables locales y globales.

b. Verdadero. La función auxiliar calcula el cuadrado de un valor. En Pascal se puede pasar una función como parámetro. Todo funciona según lo previsto.

c. Verdadero. Los únicos subrangos que se pueden declarar en Pascal son de tipo integer o de tipo char.

d. Verdadero. Se puede reescribir modificando la comparación que se hace en el repeat until. Por ejemplo...

```
var
    i : integer;
begin
    i:=0;
    repeat
        writeln("Hola");
        i:= i + 1;
    until (i = 5);
end.
```

...se puede reescribir cómo:

```
var
    i : integer;
begin
    i:=0;
    while(i <= 5) do begin
        writeln("Hola");
        i:= i + 1;
    end;
end.
```

e. Verdadero. También se pueden utilizar variables globales.

Apellido y Nombre

- Una empresa dispone de una estructura de datos con las ventas de su comercio. De cada venta se conoce número de venta, cantidad de productos y tipo de pago (efectivo o tarjeta). Se pide implementar un programa que genere una segunda estructura con las ventas cuya cantidad de productos tenga más dígitos pares que impares. En la estructura generada deben quedar almacenadas las ventas de tipo de pago efectivo antes que de tipo de pago con tarjeta. **Hecho en papel**
- Dados los siguientes programas indique para cada uno si son válidos o no. Además, analice si considera que funcionamiento en ambos programas es el mismo o no. **JUSTIFIQUE**

A	B
<pre> Program uno; Type datos = array [1..100] of integer; Var d: datos; Begin //Operaciones e invocaciones a módulos para cargar y recorrer el vector d End;</pre>	<pre> Program dos; Var d: array [1..100] of integer; Begin // Operaciones e invocaciones a módulos para cargar y recorrer el vector d End;</pre>

- Calcule e indique la cantidad de memoria estática y dinámica que utiliza el siguiente programa. **Mostrar los valores intermedios para llegar al resultado y justificar.**

<pre> program ejercicio3; type info = record nombre: string; nota: integer; datos: ^integer; end; vector = array [1..100] of info; var v: vector; i, j: integer; e: info; begin read(e.nombre); i:=0; while (i <= 100) and (e.nombre <> 'ZZZ') do begin read(e.nota); e.datos:= nil; i:= i + 1; v[i]:= e; read(e.nombre); end; for j:= 1 to i do begin new(v[j].datos); v[j].datos^:= v[j].nota MOD 10; end; end;</pre>	<table> <tr> <td>Char</td><td>1 byte</td></tr> <tr> <td>Integer</td><td>6 bytes</td></tr> <tr> <td>Real</td><td>10 bytes</td></tr> <tr> <td>Boolean</td><td>1 byte</td></tr> <tr> <td>String</td><td>Longitud + 1 byte</td></tr> <tr> <td>Puntero</td><td>4 bytes</td></tr> </table>	Char	1 byte	Integer	6 bytes	Real	10 bytes	Boolean	1 byte	String	Longitud + 1 byte	Puntero	4 bytes
Char	1 byte												
Integer	6 bytes												
Real	10 bytes												
Boolean	1 byte												
String	Longitud + 1 byte												
Puntero	4 bytes												
<p>Estática:</p> <p>e = 265b v = 26500b i = 6 j = 6</p> <p>26777 bytes</p>	<p>Dinámica:</p> <p>Suponiendo que i = 100 (valor máximo que puede alcanzar i), serían ocupados 6*100 bytes = 600 bytes.</p>												