PRÁCTICA 8

ALOCACIÓN DINÁMICA - PUNTEROS

Objetivos de la práctica:

Se espera que el alumno logre:

- **Comprender los mecanismos de alocación dinámica vs alocación estática**
- **Utilizar las operaciones *New* y *Dispose* en la resolución de problemas**

1. Escribir un pequeño programa que reserve espacio en memoria dinámica para un entero, lea su valor desde teclado, lo imprima en pantalla y luego libere la memoria alocada.

2. Dada la siguiente declaración, representar gráficamente las distintas operaciones sobre las variables de tipo puntero, e indicar el contenido de las variables asociadas y qué problemas se presentan. Represente cada variable como una caja.

```
Program punteros;

type
  ptr_bool = ^boolean;

  Ptr_compu = ^computadora;

  computadora = record
                  procesador : string;
                  memoria : integer;
                  dvd : ptr_bool;
                end;

var   c1,c2 : Ptr_compu;

begin

   a) new(c1);   new(c2);
   b) c1^.procesador := 'Intel Core i11';
   c) c2:= c1;
   d) write(c2^.procesador);
   e) write(c2^.memoria);
   f) dispose(c2);
   g) write(c2^.procesador);

end;
```

3. **Suponiendo** que Pascal reserva para los siguientes tipos de datos, la cantidad de bytes que se indica en la tabla: *(Puede variar el tamaño de acuerdo con la versión del compilador Pascal)*

Tipo	Cantidad de bytes que ocupa la representación interna en Pascal
Entero	2 bytes
Real	6 bytes
Char	1 byte
String	tantos bytes como indique la longitud del string + 1
Record	suma de las longitudes de los campos del registro
Puntero	4 bytes
Boolean	1 byte

Calcular el tamaño de memoria en los puntos señalados, suponiendo que las variables del programa ya están declaradas y se cuenta con 400.000 bytes de memoria dinámica posible de ser reservada.

```
Program Alocacion_Dinamica;
const max=100;
Type
   rango = 1..max;
   puntProducto= ^Producto;
   cadena = string[40];
   producto = record
        tipo : rango;
        descripcion : cadena;
        precio : real;
        categoria : char;
    end;

Var
p: Producto;
ptrProducto: puntProducto;

Begin
    { Suponga que en este punto hay 400.000 bytes de memoria disponible    (1) }
Readln( p.tipo );

    { Piense si la lectura anterior ha hecho variar la cantidad de memoria (2) }
New(ptrProducto);

    { ¿Cuánta memoria disponible hay ahora?                                 (3) }
Readln( ptrProducto^.tipo, ptrProducto^.descripcion );
Readln(  ptrProducto^.precio,  ptrProducto^.categoria);

    { ¿Cuánta memoria disponible hay ahora?                                 (4) }

Dispose( ptrProducto );

    { ¿Cuánta memoria disponible hay ahora?                                 (6) }
end.
```

4. Se desea ordenar en memoria una secuencia de 2.500 nombres de ciudades de longitud máxima 50 que se leen de teclado.

   a) Defina una estructura de datos estática que le permita guardar la misma información leída. Calcule su tamaño de memoria.

   b) En versiones antiguas de Pascal no era posible manejar estructuras de datos estáticas que superen los 64 Kb. Por este motivo cuando los datos a almacenar superaban este límite se debían utilizar estructuras dinámicas basadas en punteros. Si tuviéramos esta limitación para resolver a) deberíamos utilizar un vector de punteros a palabras, como se indica en la siguiente estructura:

```
Type
    NombreCiudad = String[50];
    PtrNombreCiudad = ^ NombreCiudad;
    PtrNombreCiudades = array[1..2500] of PtrNombreCiudad;
Var
    Punteros: PtrNombreCiudades;
```

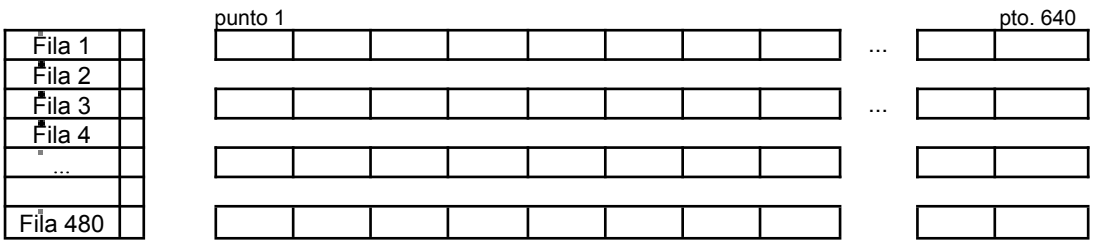
b.1.) Indique cuál es el tamaño de la variable Punteros al comenzar el programa
b.2.) Escriba un módulo que permita reservar memoria una cantidad de nombres dada (máximo 2.500). ¿Cuál es la cantidad de memoria reservada después de ejecutar el módulo para X cantidad de ciudades?
b.3.) Escriba un módulo para leer los nombres y colocarlos en la variable Punteros.

5. Dado el siguiente programa, indique que imprime:

```
Program ejemplo;

Type
      PtrInt = ^integer;

Procedure multiplica (p: PtrInt; var q: PtrInt);
Begin
q:= p;
p^  := p^  * 4;
writeln(p^);
q^  := q^  * 3;
      new(q);
q^:= 2;
writeln(p^);
End;

Var
p,q: PtrInt;

Begin
      writeln(p^);
      new (p);
      p^ := 2;
      writeln(p^);
      multiplica (p, q);
      writeln(p^);
      writeln(q^);
      dispose(p);
      dispose(q);
End.
```

6. Una imagen puede representarse como una matriz de puntos (píxeles) donde para cada uno de ellos se debe indicar su tono de gris. Dicho tono es un número entre 0 y 255, donde 0 representa negro, 255 blanco y los valores intermedios dan el tono de gris correspondiente.

Como las imágenes ocupan un gran espacio de memoria, se pensó en usar una estructura como la siguiente:



Estamos trabajando con un monitor que tiene una resolución de 480x640 píxeles (o puntos).

a) Defina la estructura de datos correspondiente.
b) Calcule el tamaño de la estructura al momento de comenzar el programa.
c) Escriba un módulo que le permita ingresar las primeras 100 filas de la pantalla (no olvide que cada fila está formada por 640 puntos). Indique el tamaño de la estructura una vez que el módulo se haya ejecutado.
d) ¿Cuál es el tamaño de la estructura completa, es decir, cuando se hayan almacenado los 480 x 640 puntos?