Práctica punteros

1. ¿Qué Se Define Como Memoria Estática?

Es aquella memoria que al asignarla cuando se declara una variable se mantiene ocupada, es decir, se define y luego no varía.

2. ¿Qué Se Define Como Memoria Dinámica?

Es un tipo de memoria el cual puede ser creado y luego destruido, es decir podemos tener un espacio de variable, usarlo y luego eliminar ese espacio o valor para utilizarlo de nuevo.

3. ¿Qué es una Variable de tipo puntero?

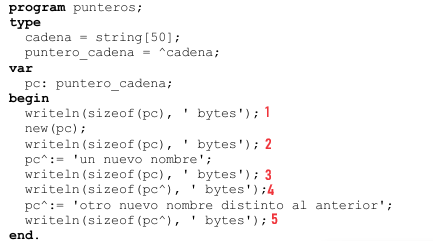
Una variable de tipo puntero es una variable que guarda una dirección de memoria y un tipo de dato, la dirección de memoria se expresa en hexadecimal, esta señala a un espacio de memoria libre, donde se puede asignar un valor del tipo de dato del puntero.

4. ¿Qué Hace La Operación “NEW”aplicada en una variable del tipo puntero?

La operación “new(puntero)” encuentra un valor de memoria disponible y lo asigna a la variable del puntero

5. ¿Qué hace la operación “DISPOSE” aplicada en una variable del tipo puntero?

Al hacer “dispose(puntero)” se borra la dirección de memoria de la variable del puntero, como también se borra el valor guardado en la memoria que previamente apuntaba el puntero, es decir, se destruye el espacio de memoria dinámica y luego desvincula al puntero de esa dirección.

PARTE PRÁCTICA

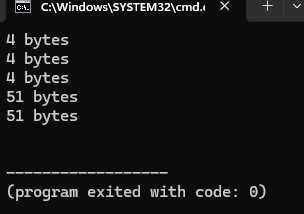
1 - imprime 4 bytes

2 - imprime 4 bytes

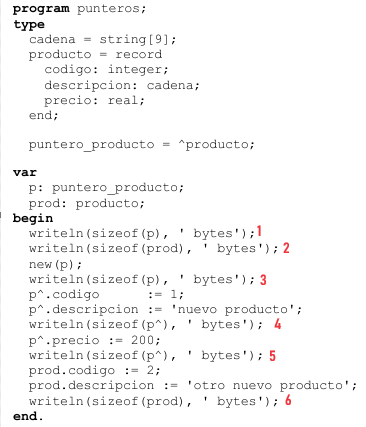
3 - imprime 4 bytes

4 - imprime 51 bytes

5 - imprime 51 bytes



2 -



1 - imprime 4 bytes

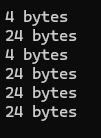
2 - imprime la suma de los campos = 2 bytes integer + 10 bytes cadena (tipos string con 9 caracteres) + 8 bytes real + 4 bytes puntero =  **imprime 24 bytes**

3 - imprime 4 bytes

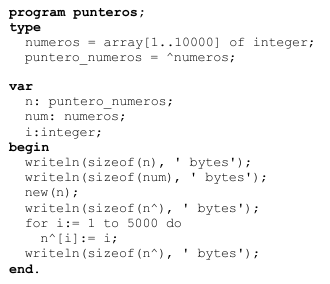
4 - imprime 24 bytes

5 - imprime 24 bytes

6 - imprime 24 bytes



3-

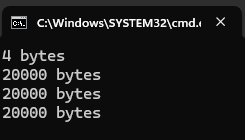


el primer write imprime 4 bytes

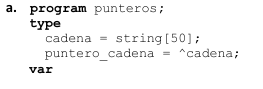
el segundo imprime 10000 x 2 bytes (tamaño de cada integer) = 20000 bytes

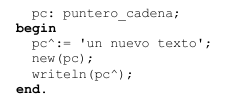
el tercero imprime 20000 bytes

el cuarto imprime 20000 bytes

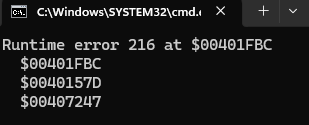


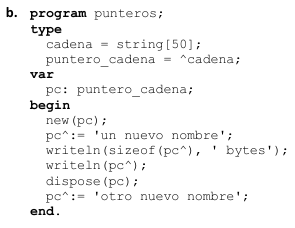
4 -





el programa imprimirá basura

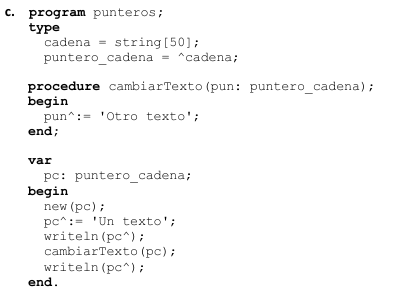




Si se conserva la última línea del código, el programa no imprimirá nada porque dará error al intentar asignar ‘un nuevo nombre’ a una dirección borrada.

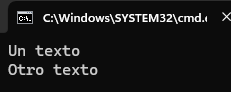
Una vez eliminada la última línea, el programa imprimirá :

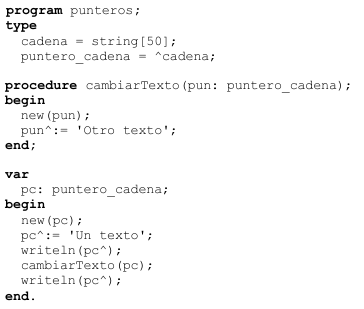
* writeln(sizeof(pc^), ‘ bytes’); imprimir 51 bytes
* writeln(pc^); imprimirá ‘Un nuevo nombre’



el primer write imprimirá ‘Un texto’

el segundo imprimirá ‘Otro texto’





El programa podría cometer errores al querer asignar una nueva dirección cuando el puntero ya tiene una asignada.

En caso que no, el programa imprimirá :

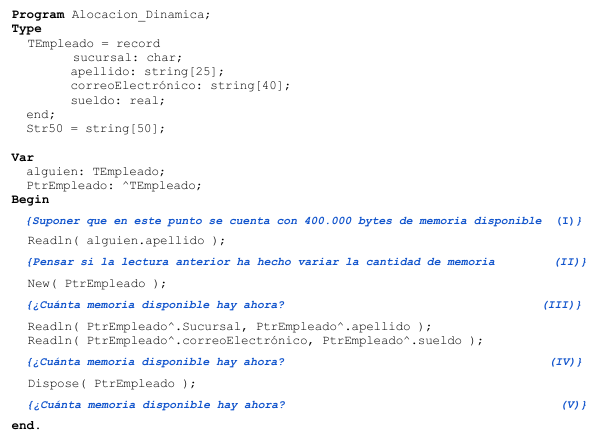
un texto

otro texto



ERROR

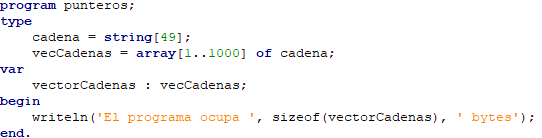
Me doy cuenta que imprime de nuevo el mismo texto porque el new(pun) asigna una dirección al puntero del procedimiento, no del código principal, por lo que se imprime 2 veces el texto del código principal.

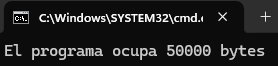


hacer

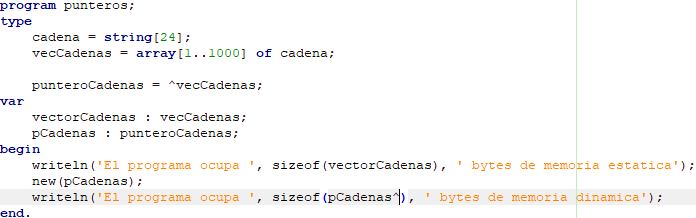
6 -

Programa que ocupe 50 KB con memoria estática



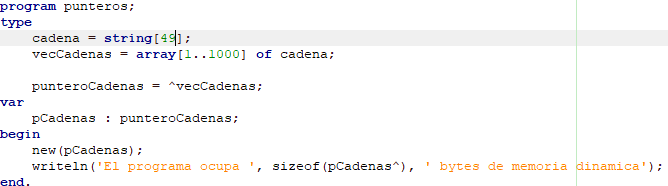


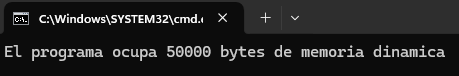
Programa que ocupa 50 KB con 50% memoria estática y 50% memoria dinámica



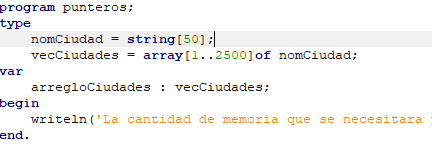


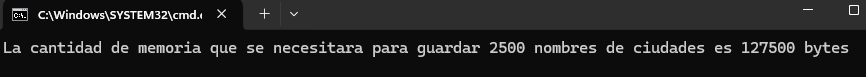
Programa que ocupa 50 KB con memoria dinámica (Hasta 4 bytes de memoria estática)





7 -

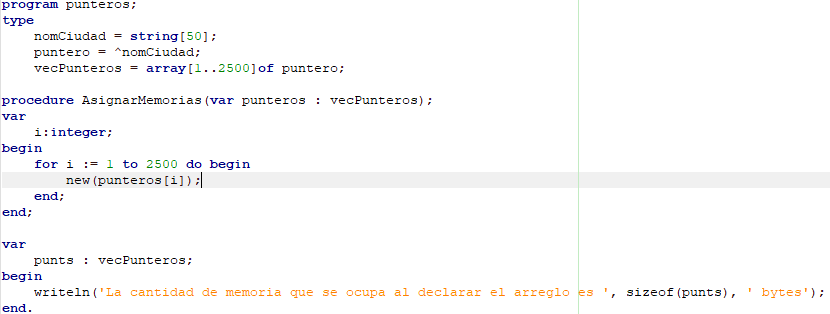






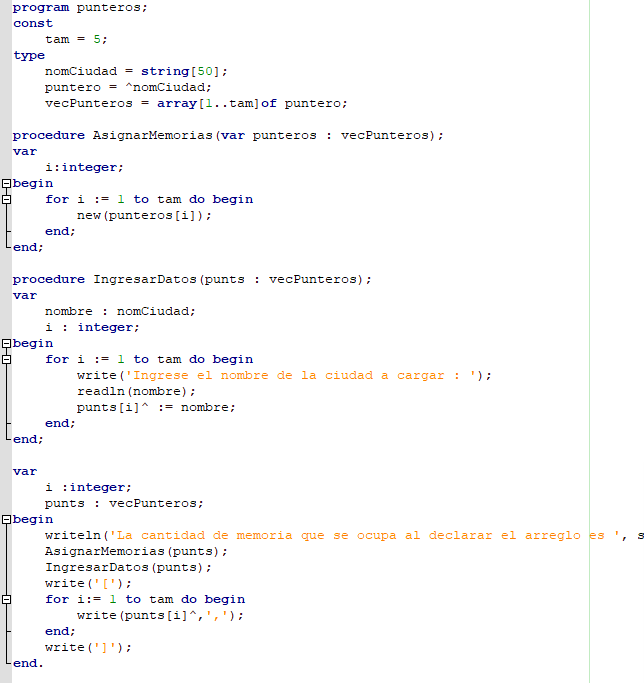
1 - La cantidad de memoria que ocupa el arreglo está determinada por = [Dim Física del arreglo x tamaño del tipo de datos ] = 2500 x 4 bytes (puntero) = 10000 bytes



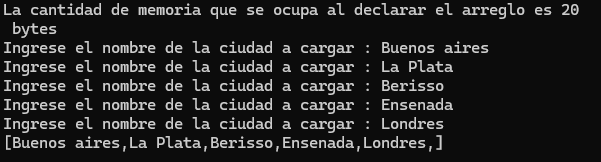
2 - 

La cantidad de memoria reservada después de ejecutar el módulo, será [memoria estática + memoria dinámica] = [(10000 bytes punteros) + (2500 espacios de memoria asignados x 51 bytes por string en cada lugar)] = 137.500 bytes

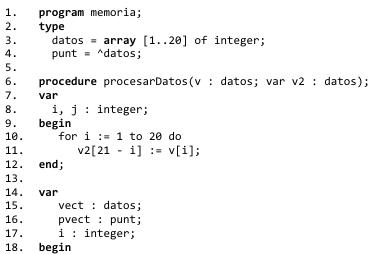
3 -



Resultado :



8 - ¿Cuánta memoria ocupa el programa?

1. 

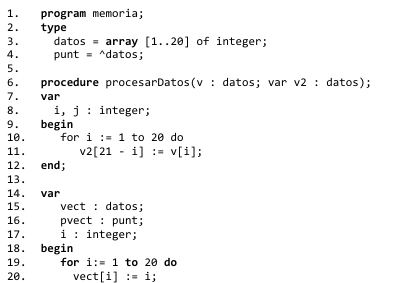
Memoria = Tamaño ‘vect’ + tamaño ‘pvect’ + tamaño ‘i’

vect = dimensión física x tamaño integer = 20 x 2 bytes = 40 bytes

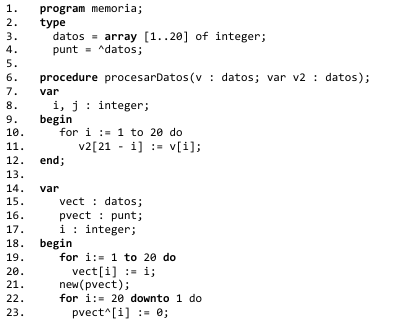
pvect = puntero = 4 bytes

i = integer = 2 bytes

Memoria = 40 + 4 + 2 = 46 bytes.

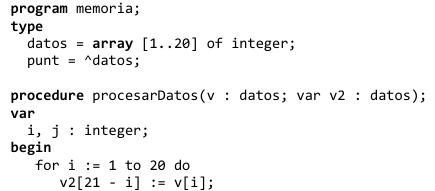
1. 

Memoria = 46 bytes

1. 

Memoria estática = 46 bytes

Memoria dinamica = 40 bytes

1. 

El procedimiento procesar Datos() utilizará una cantidad de memoria que corresponde a la sumatoria de los tamaños de los parámetros junto con los tamaños de las variables locales.

v = tipo datos = 40 bytes

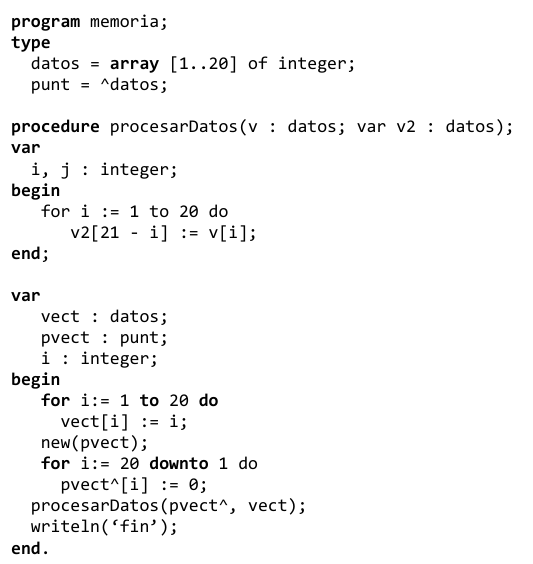
v2 = tipo puntero = 40 bytes dinámicos

i = integer = 2 bytes.

j = integer = 2 bytes.

total = 84 bytes

Es necesario aclarar que esta memoria se reserva a la hora de invocar la función, no antes, es decir, se crea dinámicamente, pero la pensamos como memoria estática.

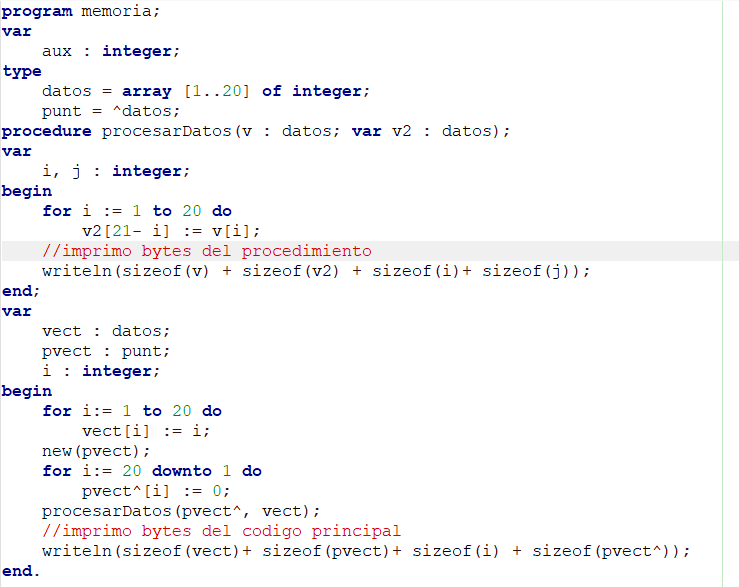
1. 

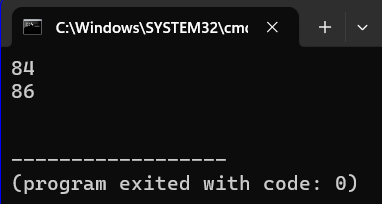
Memoria total =

Memoria vect + memoria pvect + memoria i + memoria dinámica p vect + memoria procesar Datos()

40 bytes + 4 bytes + 2 bytes + 40 bytes dinámicos + 84 bytes procesar Datos() = 170 bytes total = 130 estáticos + 40 dinámicos

Comprobación:





84 + 86 = 170 bytes