

Redictado de CADP 2020

Práctica 5 – Punteros

PARTE CONCEPTUAL

- 1) ¿Qué es la memoria estática?
- 2) ¿Qué es la memoria dinámica?
- 3) ¿Qué es una variable del tipo puntero?
- 4) ¿Qué hace la operación de NEW sobre una variable del tipo puntero?
- 5) ¿Qué hace la operación de DISPOSE sobre una variable del tipo puntero?

PARTE PRÁCTICA

Para algunos ejercicios de la parte práctica, se utilizará la función de Pascal **sizeof**, que recibe como parámetro una variable de cualquier tipo y retorna el espacio que dicha variable ocupa en la memoria principal. Para realizar estos ejercicios, considerar la siguiente tabla, que indica la cantidad de bytes que ocupa la representación interna de distintos tipos de datos en un compilador de Pascal típico.

| TIPO | CANTIDAD DE BYTES |
|---------|--|
| Entero | 2 bytes |
| Real | 4 bytes |
| Char | 1 byte |
| String | Tantos bytes como indique la longitud del String + 1 |
| Record | La suma de las longitudes de los campos del registro |
| Puntero | 4 bytes |
| Boolean | 1 byte |



- 1) Indicar los valores que imprime el siguiente programa en Pascal.

```
program punteros;
type
  cadena = string[50];
  puntero_cadena = ^cadena;
var
  pc: puntero_cadena;
begin
  writeln(sizeof(pc), ' bytes');
  new(pc);
  writeln(sizeof(pc), ' bytes');
  pc^:= 'un nuevo nombre';
  writeln(sizeof(pc), ' bytes');
  writeln(sizeof(pc^), ' bytes');
  pc^:= 'otro nuevo nombre distinto al anterior';
  writeln(sizeof(pc^), ' bytes');
end.
```

- 2) Indicar los valores que imprime el siguiente programa en Pascal.

```
program punteros;
type
  cadena = string[9];
  producto = record
    codigo: integer;
    descripcion: cadena;
    precio: real;
  end;
  puntero_producto = ^producto;
```



```

var
  p: puntero_producto;
  prod: producto;
begin
  writeln(sizeof(p), ' bytes');
  writeln(sizeof(prod), ' bytes');
  new(p);
  writeln(sizeof(p), ' bytes');
  p^.codigo := 1;
  p^.descripcion := 'nuevo producto';
  writeln(sizeof(p^), ' bytes');
  p^.precio := 200;
  writeln(sizeof(p^), ' bytes');
  prod.codigo := 2;
  prod.descripcion := 'otro nuevo producto';
  writeln(sizeof(prod), ' bytes');
end.

```

3) Indicar los valores que imprime el siguiente programa en Pascal.

```

program punteros;
type
  numeros = array[1..10000] of integer;
  puntero_numeros = ^numeros;
var
  n: puntero_numeros;
  num: numeros;
  i: integer;
begin
  writeln(sizeof(n), ' bytes');
  writeln(sizeof(num), ' bytes');
  new(n);
  writeln(sizeof(n^), ' bytes');
  for i:= 1 to 5000 do
    n^[i] := i;
  writeln(sizeof(n^), ' bytes');
end.

```

4) Indicar los valores que imprimen los siguientes programas en Pascal.

a) **program** punteros;

```

type
  cadena = string[50];
  puntero_cadena = ^cadena;
var
  pc: puntero_cadena;
begin
  pc^:= 'un nuevo texto';
  new(pc);
  writeln(pc^);
end.

```

b) **program** punteros;

```

type
  cadena = string[50];
  puntero_cadena = ^cadena;
var
  pc: puntero_cadena;
begin
  new(pc);
  pc^:= 'un nuevo nombre';
  writeln(sizeof(pc^), ' bytes');
  writeln(pc^);
end.

```

```

    dispose(pc);
    pc^:= 'otro nuevo nombre';
end.

```

```

c) program punteros;
type
    cadena = string[50];
    puntero_cadena = ^cadena;
procedure cambiarTexto(pun: puntero_cadena);
begin
    pun^:= 'Otro texto';
end;
var
    pc: puntero_cadena;
begin
    new(pc);
    pc^:= 'Un texto';
    writeln(pc^);
    cambiarTexto(pc);
    writeln(pc^);
end.

```

```

d) program punteros;
type
    cadena = string[50];
    puntero_cadena = ^cadena;
procedure cambiarTexto(pun: puntero_cadena);
begin
    new(pun);
    pun^:= 'Otro texto';
end;
var
    pc: puntero_cadena;
begin
    new(pc);
    pc^:= 'Un texto';
    writeln(pc^);
    cambiarTexto(pc);
    writeln(pc^);
end.

```

- 5) De acuerdo a los valores de la tabla que indica la cantidad de bytes que ocupa la representación interna de cada tipo de dato en Pascal, calcular el tamaño de memoria en los puntos señalados a partir de (I), suponiendo que las variables del programa ya están declaradas y se cuenta con 400.000 bytes libres.

Program Almacenamiento_Dinamico;

```

Type
    Empleado = record
        sucursal: char;
        apellido: string[25];
        correoElectronico: string[40];
        sueldo: real;
    end;
    Str50 = string[50];

```


```

Var
    alguien: Empleado;
    PtrEmpleado: ^Empleado;

```



Begin

```
{Suponer que en este punto se cuenta con 400.000 bytes de memoria disponible (I) }
Readln( alguien.apellido );
{Pensar si la lectura anterior ha hecho variar la cantidad de memoria (II) }
New( PtrEmpleado );
{¿Cuánta memoria disponible hay ahora? (III) }
Readln( PtrEmpleado^.Sucursal, PtrEmpleado^.apellido );
Readln( PtrEmpleado^.correoElectrónico, PtrEmpleado^.sueldo );
{¿Cuánta memoria disponible hay ahora? (IV) }
Dispose( PtrEmpleado );
{¿Cuánta memoria disponible hay ahora? (V) }
end.
```

- 6) Se desea almacenar en memoria una secuencia de 2500 nombres de ciudades, cada nombre podrá tener una longitud máxima 50 caracteres. 

- a) Definir una estructura de datos estática que permita guardar la información leída. Calcular el tamaño de memoria que requiere la estructura.
- b) Dado que Pascal no permite manejar estructuras de datos estáticas que superen los 64 Kb, pensar en utilizar un vector de punteros a palabras, como se indica en la siguiente estructura:

```
Type  Nombre = String[50];
        Puntero = ^Nombre;
        ArrPunteros = array[1..2500] of Puntero;
Var
        Punteros: ArrPunteros;
```

- b.1) Indicar cuál es el tamaño de la variable Punteros al comenzar el programa. 
- b.2) Escribir un módulo que permita reservar memoria para los 2.500 nombres. ¿Cuál es la cantidad de memoria reservada después de ejecutar el módulo? 
- b.3) Escribir un módulo para leer los nombres y almacenarlos en la estructura de la variable Punteros.

