Juan Menduiña

Trabajo Práctico N° 5:
Punteros.

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 la cantidad de bytes que dicha variable ocupa en la memoria principal. Se presenta 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. Se recomienda graficar cada una de las situaciones planteadas a partir de una prueba de escritorio.

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

Tabla de referencia de tamaño de los tipos de datos de Pascal (estos valores pueden variar entre diferentes implementaciones del compilador)

Ejercicio 1.

Indicar los valores que imprime el siguiente programa en Pascal. Justificar mediante prueba de escritorio.

```
program TP5_E1;
{$codepage UTF8}
uses crt;
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.
```

Instrucción	pc	pc^	write
`writeln(sizeof(pc),' bytes');`			4 bytes
`new(pc);`	XXX		
`writeln(sizeof(pc),' bytes');`			4 bytes
`pc^:='un nuevo nombre';`		‘un nuevo nombre’	
`writeln(sizeof(pc),' bytes');`			4 bytes
`writeln(sizeof(pc^),' bytes');`			51 bytes
`pc^:='otro nuevo nombre distinto al anterior';`		‘otro nuevo nombre distinto al anterior’	
`writeln(sizeof(pc^),' bytes');`			51 bytes

Ejercicio 2.

Indicar los valores que imprime el siguiente programa en Pascal. Justificar mediante prueba de escritorio.

```
program TP5_E2;
{$codepage UTF8}
uses crt;
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.
```

Instrucción	p	p^	prod	write
`writeln(sizeof(p),' bytes');`				4 bytes
`writeln(sizeof(prod),' bytes');`				16 (24) bytes
`new(p);`	XXX			
`writeln(sizeof(p),' bytes');`				4 bytes
`p^.codigo:=1;`		.codigo=1		
`p^.descripcion:='nuevo producto';`		.descripcion=’nuevo pro’		warming
`writeln(sizeof(p^),' bytes');`				16 (24) bytes
`p^.precio:=200;`		.precio=200		
`writeln(sizeof(p^),' bytes');`				16 (24) bytes
`prod.codigo:=2;`			.codigo=2	
`prod.descripcion:='otro nuevo producto';`			.descripcion=’otro nuev’	warming
`writeln(sizeof(prod),' bytes');`				16 (24) bytes

Ejercicio 3.

Indicar los valores que imprime el siguiente programa en Pascal. Justificar mediante prueba de escritorio.

```
program TP5_E3;
{$codepage UTF8}
uses crt;
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.
```

Instrucción	n	n^	num	i	write
`writeln(sizeof(n),' bytes');`					4 bytes
`writeln(sizeof(num),' bytes');`					20000 bytes
`new(n);`	XXX				
`writeln(sizeof(n^),' bytes');`					20000 bytes
`for i:= 1 to 5000 do n^[i]:=i;`			n^[1..5000]=[1..5000]	[1..5000]	
`writeln(sizeof(n^),' bytes');`					20000 bytes

Ejercicio 4.

Indicar los valores que imprime los siguientes programas en Pascal. Justificar mediante prueba de escritorio.

(a)

```
program TP5_E4a;
{$codepage UTF8}
uses crt;
type
  cadena=string[50];
  puntero_cadena=^cadena;
var
  pc: puntero_cadena;
begin
  pc^:='un nuevo texto';
  new(pc);
  writeln(pc^);
end.
```

Instrucción	pc	pc^	write
`pc^:='un nuevo texto';`		error	
`new(pc);`	XXX		
`writeln(pc^);`			basura

(b)

```
program TP5_E4b;
{$codepage UTF8}
uses crt;
type
  cadena=string[50];
  puntero_cadena=^cadena;
var
  pc: puntero_cadena;
begin
  new(pc);
  pc^:='un nuevo nombre';
  writeln(sizeof(pc^),' bytes');
  writeln(pc^);
  dispose(pc);
  pc^:='otro nuevo nombre';
end.
```

Instrucción	pc	pc^	write
`new(pc);`	XXX		
`pc^:='un nuevo nombre';`		‘un nuevo nombre’	
`writeln(sizeof(pc^), ' bytes');`			51 bytes
`writeln(pc^);`			un nuevo nombre
`dispose(pc);`	elimina puntero y libera memoria		
`pc^:='otro nuevo nombre';`		error	

(c)

```
program TP5_E4c;
{$codepage UTF8}
uses crt;
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.
```

Instrucción	pc	pc^	write
`new(pc);`	XXX		
`pc^:='Un texto';`		'Un texto'	
`writeln(pc^);`			Un texto
`cambiarTexto(pc);`		'Otro texto'	
`writeln(pc^);`			Otro texto

(d)

```
program TP5_E4d;
{$codepage UTF8}
uses crt;
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.
```

Instrucción	pc	pc^	copia pc	copia pc^	write
`new(pc);`	XXX				
`pc^:='Un texto';`		'Un texto'			

`writeln(pc^);`					Un texto
`cambiarTexto(pc);`			YYY	‘Otro texto’	
`writeln(pc^);`					Un texto

Ejercicio 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. Justificar mediante prueba de escritorio.

```
program TP5_E5;
{$codepage UTF8}
uses crt;
type
  TEmpleado=record
    sucursal: char;
    apellido: string[25];
    correoElectronico: string[40];
    sueldo: real;
  end;
  Str50=string[50];
var
  alguien: TEmpleado;
  PtrEmpleado: ^TEmpleado;
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^.correoElectronico,PtrEmpleado^.sueldo);
  {¿Cuánta memoria disponible hay ahora? (IV)}
  dispose(PtrEmpleado);
  {¿Cuánta memoria disponible hay ahora? (V)}
end.
```

Instrucción	Memoria
`readln(alguien.apellido);`	400.000 bytes
`new(PtrEmpleado);`	399.928 (399.924) bytes (400.000 - 72)
`readln(PtrEmpleado^.Sucursal,PtrEmpleado^.apellido);`	399.928 (399.924) bytes
`readln(PtrEmpleado^.correoElectrónico,PtrEmpleado^.sueldo);`	399.928 (399.924) bytes
`dispose(PtrEmpleado);`	400.000 bytes (399.928 + 72)

Ejercicio 6.

Realizar un programa que ocupe 50 KB de memoria en total. Para ello:

(a) *El programa debe utilizar sólo memoria estática.*

```
program TP5_E6a;
{$codepage UTF8}
uses crt;
const
  KB=50; byte=1024; bytes=KB*byte;
  vector_total=bytes div 2;
type
  t_vector=array[1..vector_total] of int16;
var
  vector: t_vector;
begin
  textcolor(green); write('La memoria estática ocupada por el vector es '); textcolor(red);
write(sizeof(vector)/byte:0:2); textcolor(green); write(' KB');
end.
```

(b) *El programa debe utilizar el 50% de memoria estática y el 50% de memoria dinámica.*

```
program TP5_E6b;
{$codepage UTF8}
uses crt;
const
  KB=50; byte=1024; bytes=KB*byte;
  vector_total=(bytes div 2)-2;
type
  t_vector=array[1..vector_total] of int16;
  t_string=string[3];
  t_registro_vector=record
    vector: t_vector;
    palabra: t_string;
  end;
  t_puntero_registro=^t_registro_vector;
var
  vector: t_vector;
  puntero_registro: t_puntero_registro;
begin
  new(puntero_registro);
  textcolor(green); write('La memoria estática ocupada por vector es '); textcolor(red);
write(sizeof(vector)/byte:0:8); textcolor(green); writeln(' KB');
  textcolor(green); write('La memoria estática ocupada por puntero_registro es ');
textcolor(red); write(sizeof(puntero_registro)/byte:0:8); textcolor(green); writeln(' KB');
  textcolor(green); write('La memoria estática ocupada total es '); textcolor(red);
write(sizeof(vector)/byte+sizeof(puntero_registro)/byte:0:8); textcolor(green); writeln('
KB');
  textcolor(green); write('La memoria dinámica ocupada por contenido puntero_registro es ');
textcolor(red); write(sizeof(puntero_registro^)/byte:0:8); textcolor(green); write(' KB');
end.
```

(c) *El programa debe minimizar tanto como sea posible el uso de la memoria estática (a lo sumo, 4 bytes).*

```
program TP5_E6c;
{$codepage UTF8}
uses crt;
const
  KB=50; byte=1024; bytes=KB*byte;
  vector_total=(bytes div 2)-2;
type
  t_vector=array[1..vector_total] of int16;
  t_puntero_vector=^t_vector;
var
  puntero_vector: t_puntero_vector;
begin
  new(puntero_vector);
  textcolor(green); write('La memoria estática ocupada por puntero_vector es ');
textcolor(red); write(sizeof(puntero_vector)/byte:0:8); textcolor(green); writeln(' KB');
  textcolor(green); write('La memoria dinámica ocupada por contenido puntero_vector es ');
textcolor(red); write(sizeof(puntero_vector^)/byte:0:8); textcolor(green); write(' KB');
end.
```

Ejercicio 7.

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

(a) *Definir una estructura de datos estática que permita guardar la información leída. Calcular el tamaño de memoria que requiere la estructura.*

```
program TP5_E7a;
{$codepage UTF8}
uses crt;
const
  longitud_ciudad=50;
  ciudades_total=2500;
type
  t_ciudad=string[longitud_ciudad];
  t_vector_ciudad=array[1..ciudades_total] of t_ciudad;
begin
  textcolor(green); write('El tamaño de memoria que requiere la estructura es ');
textcolor(red); write(sizeof(t_vector_ciudad)); textcolor(green); write(' bytes');
end.
```

(b) *Dado que un compilador de Pascal típico 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;

(i) *Indicar cuál es el tamaño de la variable Punteros al comenzar el programa.*

El tamaño de la variable *"Punteros"*, al comenzar el programa, es 10.000 bytes.

(ii) *Escribir un módulo que permita reservar memoria para los 2500 nombres. ¿Cuál es la cantidad de memoria reservada después de ejecutar el módulo? ¿La misma corresponde a memoria estática o dinámica?*

La cantidad de memoria reservada después de ejecutar el módulo es 127.500 bytes, la cual corresponde a memoria dinámica.

(iii) *Escribir un módulo para leer los nombres y almacenarlos en la estructura de la variable Punteros.*

```
program TP5_E7b;
{$codepage UTF8}
uses crt;
const
  longitud_ciudad=50;
  ciudades_total=2500;
type
  t_ciudad=string[longitud_ciudad];
  t_puntero_ciudad=^t_ciudad;
  t_vector_ciudad=array[1..ciudades_total] of t_puntero_ciudad;
procedure reservar_memoria(var vector_ciudad: t_vector_ciudad);
var
  i: int16;
begin
  for i:= 1 to ciudades_total do
    new(vector_ciudad[i]);
end;
function random_string(length: int8): string;
var
  i: int8;
  string_aux: string;
begin
  string_aux:='';
  for i:= 1 to length do
    string_aux:=string_aux+chr(ord('A')+random(26));
  random_string:=string_aux;
end;
procedure leer_ciudades(var vector_ciudad: t_vector_ciudad);
var
  i: int16;
begin
  for i:= 1 to ciudades_total do
    vector_ciudad[i]^:=random_string(5+random(6));
end;
var
  vector_ciudad: t_vector_ciudad;
  i: int16;
begin
  randomize;
  for i:= 1 to ciudades_total do
    vector_ciudad[i]:=nil;
  textcolor(green); write('El tamaño de la variable vector_ciudad es '); textcolor(red);
write(sizeof(vector_ciudad)); textcolor(green); writeln(' bytes');
  textcolor(green); write('El tamaño del contenido de la variable vector_ciudad es ');
textcolor(red); write(sizeof(vector_ciudad[1]^)*length(vector_ciudad)); textcolor(green);
writeln(' bytes');
  reservar_memoria(vector_ciudad);
  textcolor(green); write('El tamaño de la variable vector_ciudad es '); textcolor(red);
write(sizeof(vector_ciudad)); textcolor(green); writeln(' bytes');
  textcolor(green); write('El tamaño del contenido de la variable vector_ciudad es ');
textcolor(red); write(sizeof(vector_ciudad[1]^)*length(vector_ciudad)); textcolor(green);
writeln(' bytes');
  leer_ciudades(vector_ciudad);
  textcolor(green); write('El tamaño de la variable vector_ciudad es '); textcolor(red);
write(sizeof(vector_ciudad)); textcolor(green); writeln(' bytes');
  textcolor(green); write('El tamaño del contenido de la variable vector_ciudad es ');
textcolor(red); write(sizeof(vector_ciudad[1]^)*length(vector_ciudad)); textcolor(green);
write(' bytes');
  for i:= 1 to ciudades_total do
    dispose(vector_ciudad[i]);
end.
```

Ejercicio 8.

Analizar el siguiente programa:

```
program TP5_E8;
{$codepage UTF8}
uses crt;
type
  datos=array[1..20] of integer;
  punt=^datos;
procedure procesarDatos(v: datos; var v2: datos);
var
  i, j: integer;
begin
  for i:= 1 to 20 do
    v2[21-i]:=v[i];
end;
var
  vect: datos;
  pvect: punt;
  i: integer;
begin
  for i:= 1 to 20 do
    vect[i]:=i;
  new(pvect);
  for i:= 20 downto 1 do
    pvect^[i]:=0;
  procesarDatos(pvect^,vect);
  writeln('fin');
end.
```

Responder: ¿Cuánta memoria en total ocupa el programa al ejecutarse? Considerar tanto variables estáticas como dinámicas, parámetros y variables locales de los módulos.

Hasta sentencia de la línea	Memoria estática	Memoria dinámica	Memoria total
(a) 18	46 bytes	0 bytes	46 bytes
(b) 20	46 bytes	0 bytes	46 bytes
(c) 23	46 bytes	40 bytes	86 bytes
(d) 11	0 bytes	0 bytes	0 bytes
(e) 25	46 bytes	40 bytes	86 bytes

Aclaración: Hasta la sentencia de la línea 24, tenemos 88 bytes en memoria dinámica, ya que se suman 40 bytes por el parámetro por valor, 4 bytes por el parámetro por referencia y 4 bytes por las variables locales al proceso *"procesarDatos"*.