UNIVERSIDAD CENTRAL DE VENEZUELA
FACULTAD DE CIENCIAS
ESCUELA DE COMPUTACIÓN
ALGORITMOS Y ESTRUCTURAS DE DATOS

PRÁCTICA DEL TIPO DE DATO APUNTADOR

1. ¿Cuál es la diferencia entre los operadores ref y *?
2. ¿Es posible tener apuntadores de apuntadores y anidarlos cuantas veces sea necesario?¿Cuál sería su utilidad?
3. Para las siguientes instrucciones, construya el estado de todas las variables en la memoria (de forma gráfica) que muestre que ocurre en ella, y cuál es la salida del programa.

```
Register rNode
  Integer iInfo
  rNode* pNext
end
```

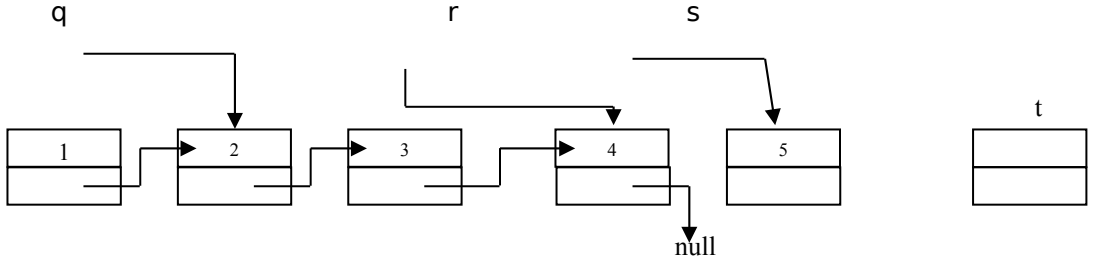
```
1  void main()                 8   *s.pNext = p
2     rNode* p, r, s           9   *s.iInfo = 3
3     p = new rNnode           10   (*(*(*p.pNext).pNext).pNext).iInfo = 2
4     s = new rNnode           11   (*(*s.pNext).pNext).iInfo = 1
5     r = new rNnode           12   p = *s.pNext
6     *p.pNext = r             13   Print( *p.iInfo +" "+ *s.iInfo +" "+ *r.iInfo);
7     *r.pNext = s             14  end
```

4. Dada las siguiente definiciones:

```
class Nodo
public:
   Integer N
   Nodo* prox
end
Nodo *q, r, s
Nodo t
```

Asuma como estado inicial la figura que a continuación se muestra, e indique el estado final después de ejecutar cada una de las siguientes instrucciones independientemente y luego una tras otra secuencialmente.

q = *q.prox	*s.prox= s
*q = *(*q.prox)	t = *q
*q.prox	*q = *s
*(*q.prox).prox	*s = t
q = *r.prox	q = t
*q = *(*r.prox)	*r.prox =q
*s.prox = *q.prox	q= *(*(*q.prox).prox).prox



5. Dada la siguiente secuencia de instrucciones indique que ocurre en c/línea. Indique si queda algún espacio de memoria por liberar.

```
class Nodo
public:
    Integer Info
    Nodo* prox
end
Nodo* P, Q
Integer* E
Integer I
Integer** F

void theFunction()
    P = New Nodo
    E = New Integer
    *E = 0;
    F = ref E
```

```
        *P. prox = New Nodo
        Q = ref (*P)
        P = *P.prox
        *Q.Info = 30
        *P. Info = *Q.Info + 10
        *P.prox = New Nodo
        *(*P.prox).Info = *Q.Info + *P.Info + 10
        P = *P.prox
        *P.prox = NIL
        while Q !=  NIL do
            *E = *F + *Q.Info
            Q = *Q.prox
        end
        F = new Integer*
        *F = new Integer
        **F= 1
        delete *F
        *F = ref I
        I = 5
        Print(**F)
    end
```

6. Considere las siguientes declaraciones:

```
Integer* X, Y, Z
Char* W
Integer A
Char B
Boolean C
```

Indique el efecto de las siguientes operaciones:

```
1   X = new Integer      9   X = new Integer
2   Y = new Integer      10  *X = 1
3   W = new Char         11  *W = 'G'
4   X= Y                 12  A = *X + *Y
5   B = *W               13  C = (*W == A)
6   Z = new Integer      14  *Z = A
7   Z = W                15  Z= X
8   C= (W==Z)            16  delete Y
```

7. Dada la siguiente secuencia de instrucciones indique que ocurre en cada línea. Indique además si queda algún elemento por liberar de memoria al terminar LOL( ).

```
class Nodo
public:
        Integer Info
        Nodo* prox
end
Nodo* P, Q

void LOL( )
        P = New Nodo
        *P.Info = 10
        *P.prox = P
        Q = New Nodo
        *P.prox = Q
        *Q.Info = *P.Info + 3
        *Q.prox = P
        *Q.prox = NULL
        Q = New Nodo()
        *Q.Info = *P.Info + *(*P.prox).Info
        *(*P.prox).prox = Q
        *Q.prox = NIL
        Q = P
```

```
        while Q != NULL do
            Print (*Q.Info)
            Q = *Q.prox
        end
    end
```

8. Haga la traza del siguiente conjunto de instrucciones y explique qué sucede en cada instrucción.

```
Integer i, j, n
Integer* arr
Integer** mat
Read(n)
arr = new Integer[n]
mat = new Integer*[n]
1   for i=1 to n do
2       arr[i]= i
3       mat[i]=new Integer[n]
4       for j=1 to n do
5           mat[i][j]=i+j
6       end
7   end
8   i=n-1
9   while i>=0 do
10      Print(arr[i])
11      delete []mat[i]
12      i=i-1
13  end
14  delete []arr
15  delete []mat
16  // en este momento, ¿a quién apunta arr?.
    ¿Podría acceder arr[1]?
```

9. Realice la traza del siguiente algoritmo. En cada línea, muestre el estado de las estructuras de datos gráficamente, y en caso de que la instrucción sea incorrecta, indicar el tipo de error.

```
Register Nodo
    Integer Info
    Nodo* prox
end

Nodo** p, s
Nodo* q, r
1   p = ref q
2   *q.info = 30
3   *q.prox = NIL
4   r = new Nodo
5   *r.info = 2
6   s = ref r
7   delete q
8   q = new Nodo; *q.info=31
9   *s.prox = q
10  **p.info = *q.info + *(**s.prox).info
```

10. Haga la traza del siguiente algoritmo y explique que sucede en cada instrucción.

```
class Point
public:
    Real x,y,
    Constructor Point()
        x=0
        y=0
    end
    Constructor Point(Real x, Real y)
        *this.x=x
        *this.y=y
    end
end
class Rect
public:
    Point* p1,p2
    Constructor Rect()
        p1=NIL
        p2=NIL
    end

    Constructor Rect(Point* p1, Point* p2)
        *this.p1 = p1
        *this.p2 = p2
    End
```

```
        Destructor Rect()
                if p1 != NULL then
                        delete p1
                end
                if p2 != NULL then
                        delete p2
                end
        end
    end
  void main()
        Point* a, b
        a = new Point
        b = new Point(1,1)
        Rect* r = new Rect(a,b)
        delete a
        delete r  // ¿Qué error ocurre aquí?
        Rect otro(a,b)
  end   // ¿Qué sucede al llamar al destructor de otro?
```

GDAYED/2014