1.5. TDA Arreglos

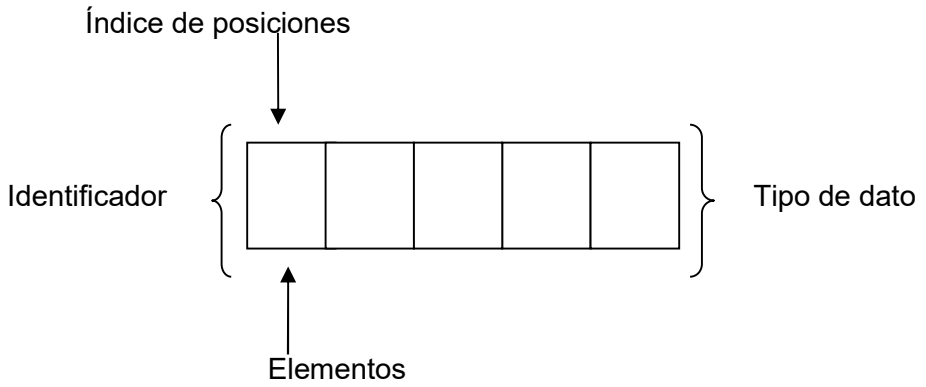
Estructura estática o fundamental, de tipo lineal, los arrays se definen como un área continua de memoria de elementos del mismo tamaño cada uno de estos elementos indexados continuamente. Lo especial en los arreglos es que se tiene un acceso aleatorio (tiempo de acceso constante a cualquier elemento en un array)

*Tiempo constante en Lectura

*Tiempo constante en Escritura

El arreglo es una estructura de acceso aleatorio, porque todos los componentes pueden seleccionarse al azar y son igualmente accesibles rápidamente [8]. Si se quiere acceder a la dirección de un elemento i se puede aplicar la fórmula aritmética [9]:

array_addr + elem_size × (i − first_index)



Cuando se crea un arreglo, se reserva un espacio de memoria, se utilice o no.

1.5.1. Operaciones Básicas

- Insertar
- Eliminar
- Modificar
- Ordenar
- Búsqueda

Tamaño = Ind Sup - Ind Inf + 1

1.5.2. Arreglo Unidimensional

Metodología Estructurada.
INSERTAR

```
//declaraciones generales
Constante max 10
opc: bolean =verdadero
1.  a=arreglo [max]  : enteros
2.  ind, dato, i  : entero =0
3. función InsertarElemento (entero):boolean
4. función EliminaDesorden (entero):boolean
5. función ImprimeArreglo():cadena
6. función BuscaSec(entero): entero

//definición de funciones
7. función InsertarElemento (entero d):boolean
// se inserta si hay espacio y si es el mismo tipo de dato

  a.  si  ind<max y d>0 entonces
       a.1    a[ind]=d
       a.2    incrementar ind+1
       a.3    retorna true
   b. si no
            retorna false
   fin 7.a.
fin 7
8. función EliminaDesorden (entero d)  :boolean
     a.  pos:entero = 0
     b.   si  (ind>0) entonces
     c.       pos=BuscaSec(d)
     d.       ind=ind-1;
     e.   repetir desde i=pos hasta ind incremento de i
              e1.   a[i]=a[i+1];
          fin e
     f.   a[ind]=0
     g.  retorna verdadero
        si no
            retorna falso
     h.  fin b
fin 8
9. función BuscaSec(entero d): entero
   a. j:entero=0
     b. repetir mientras (j<ind) y (a[j]!=d) hacer
          j=j+1
      fin b.
        retorna j
fin 9
10.  función ImprimeArreglo():cadena
     a.    value: cadena=nulo
     b.    repetir desde i=0 hasta max
     c.    value=value + a[i]//concatena los valores
     d.    retorna value
fin 10.
```

```
Función void main ()
1.    mientras opc=verdadero
```

```
        a.  imprimir (“dame el dato a insertar”)
        b.  lee(dato)
        c.  si InsertarElemento (dato)=true entonces
              i.  imprimir (“se insertó el elemento”)
        d.  si no
              i.  imprimir (“no se inserto elemento”)
            fin c.
        e.  imprimir (“Desea insertar otro elemento”)
        f.  leer (opc)
        fin 1.
2.   imprimir(a)
     //elimina un elemento del arreglo
3.   imprimir (“dame el elemento a eliminar”)
8.    Lee(dato)
9.   Si (EliminaDesorden(dato))=verdadero entonces
10.       Imprimi(“si elimino”)
11.  Si no
12        Imprimir(“No eliminó”)
13    fin 9.
     Imprimir(a)
fin 1.
```

Ejercicio 3.- De forma individual codificar las funciones de Arreglo Unidimensional, realice el TDArreglo.

*Especificación

*Niveles de abstracción

*Implementación

1.5.3. Arreglo Unidimensional con inserción ordenada.

ESTRUCTURADA

INSERTAR
//declaraciones generales
1. Constante max 10
2. opc: bolean =verdadero
3. a=arreglo [max] : enteros
4. ind, dato,pos : entero =0
5. función InsertarOrdenado (entero):boolean
6. función EliminaOrdenado (entero):boolean
7. función ImprimeArreglo():cadena
8. función BuscaSec(entero): entero
 	
//definición de funciones

```
9. función BuscaSec(datob:entero):entero
    a.  j:entero=0
     b.  repetir mientras (j<ind) y datob>a[j] hacer
         b1.  j=j+1
     fin b.
     c. si (j>=ind) o (datob<a[j]) entonces
         retorno -j
     si no
        retorno j
 fin 9.

10. funcion InsertarOrdenado (dato:entero): boolean
      a. si (ind<max) entonces
          a1.  pos=busqueda_sec(dato)
           a2.  si (pos>0) entonces
                 retorno falso
           a3. si no
           a4.    pos=pos*(-1)
           a5.  repetir desde  i=ind hasta i>= pos+1 en decremento de i hacer
                  a[i]=a[i-1]
                fin a5.
                 a[pos]=dato
                 hacer ind=ind+1
                 retorno verdadero
          fin a3.
      fin a.
      b si no
          retorna falso //no hay espacio
 fin 10.
11. Función EliminaOrdenado(dato:entero):boolean
       a.  pos=0
      b. si (ind>0)
      c. pos = BuscaSec (dato)
      d. si (pos<0)
            retorna falso //elemento no existe
      e. si no
          e1.  ind=ind-1
          e2.   repetir desde i=pos hasta ind en incremento de i hacer
           e3.       a[i]=a[i+1]
          e4    fin e2.
          e5.      Retorna verdadero// se elimino
        f. fin e.
     g.  si no
          retorna falso //esta vacio el arreglo
   fin 11.
12.  función ImprimeArreglo():cadena
      a.    value: cadena=nulo
      b.    repetir desde i=0 hasta max
      c.    value=value + a[i]//concatena los valores
      d.    retorna value
fin 12.
```

```
1. función main():void
    a.  mientras (opc=verdadero)
         a1   imprimir ("Dame el elemento a insertar")
         a2.  leer(dato)
         a3.  si (InsertarOrdenado (dato))= verdadero
                  imprimir("Se insertó el elemento")
              si no
                  imprimir ("No se inserto")
         a4. ("Deseas seguir insertando si 1 no 0")
         a5   leer(opc)
     fin a.
      imprimir(ImprimeArreglo ())
//elimina un elemento del arreglo
2. Imprimir ("dame el elemento a eliminar")
3. Lee(dato)
4.   Si (EliminaOrdenado (dato))=verdadero entonces
       Imprimi("si elimino")
    Si no
      Imprimir("No eliminó")
6.   fin 9.
 7.    Imprimir(ImprimeArreglo())
fin 1.
```

Ejercicio 4.- De forma individual codificar las funciones de Arreglo Unidimensional, realice el TDArregloOrdenado que albergue n elementos de números enteros positivos y negativos.

*Especificación

*Niveles de abstracción

*Implementación.

1.6. Arreglos Bidimensionales o Matriz

En una estructura de datos homogénea, finita, ordenada, se hace referencia a sus elementos por medio de índices, las dimensiones de los arreglos dependen del lenguaje de programación donde se emplean.

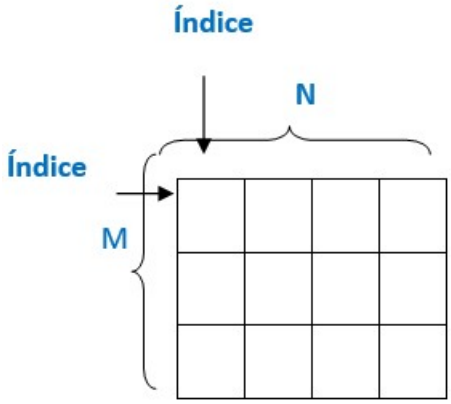
El arreglo bidimensional utiliza 2 índices para hacer referencia a sus elementos.

Arreglo = Array [índiceinf - indicesup , indiceinf - indicesup] tipo de dato

M N

Tamaño arreglo = M*N

Representación de un arreglo bidimensional.



Calcular el tamaño

NTC = [(Limsup - liminf + 1) * (limsup - liminf + 1)]

(3 - 1 + 1) * (4 - 1 + 1)

3 * 4 = 12

Calcular el tamaño total del arreglo bidimensional

arreglo [‘a’ . . . . ‘z’ . . . . ., ‘-5’. . . . . ‘5’]

arreglo [(27 - 1 + 1) * (5 + 5 + 1)]

arreglo = 27 * 11

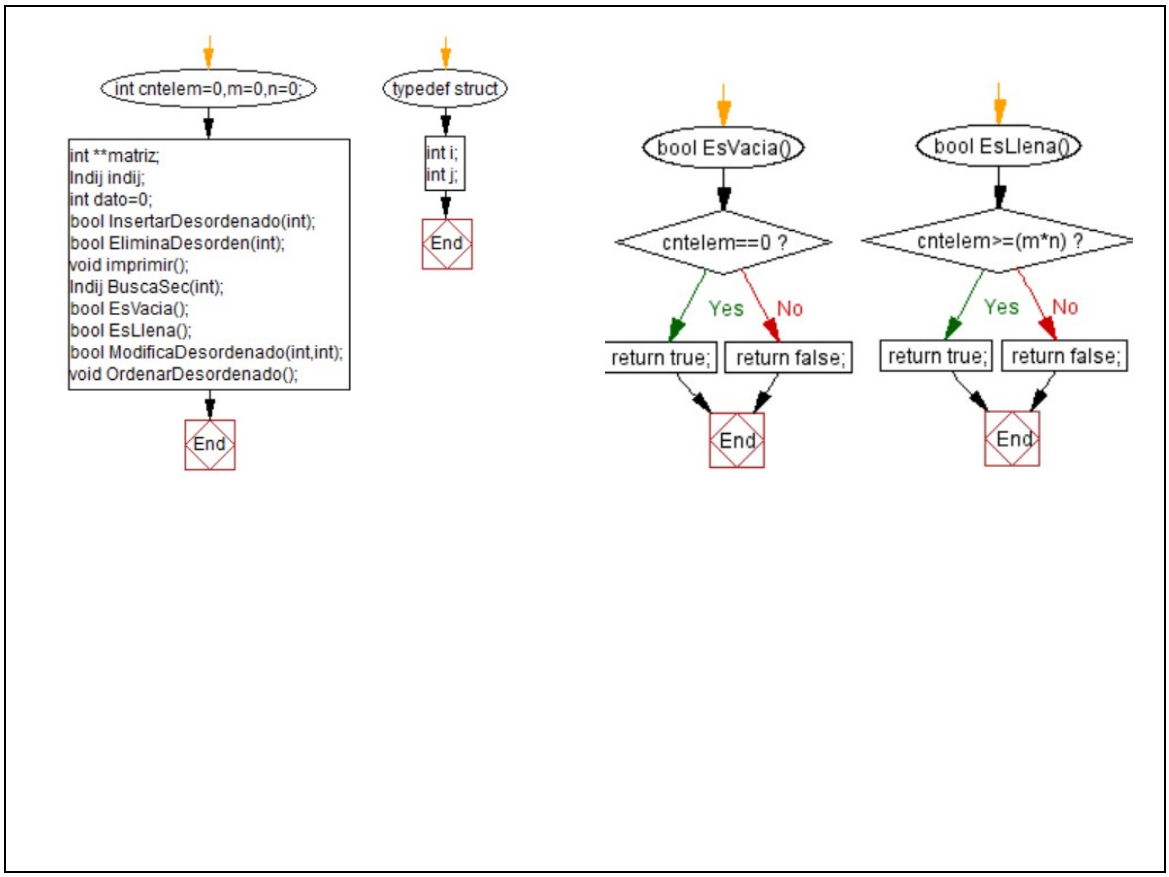
arreglo = 297

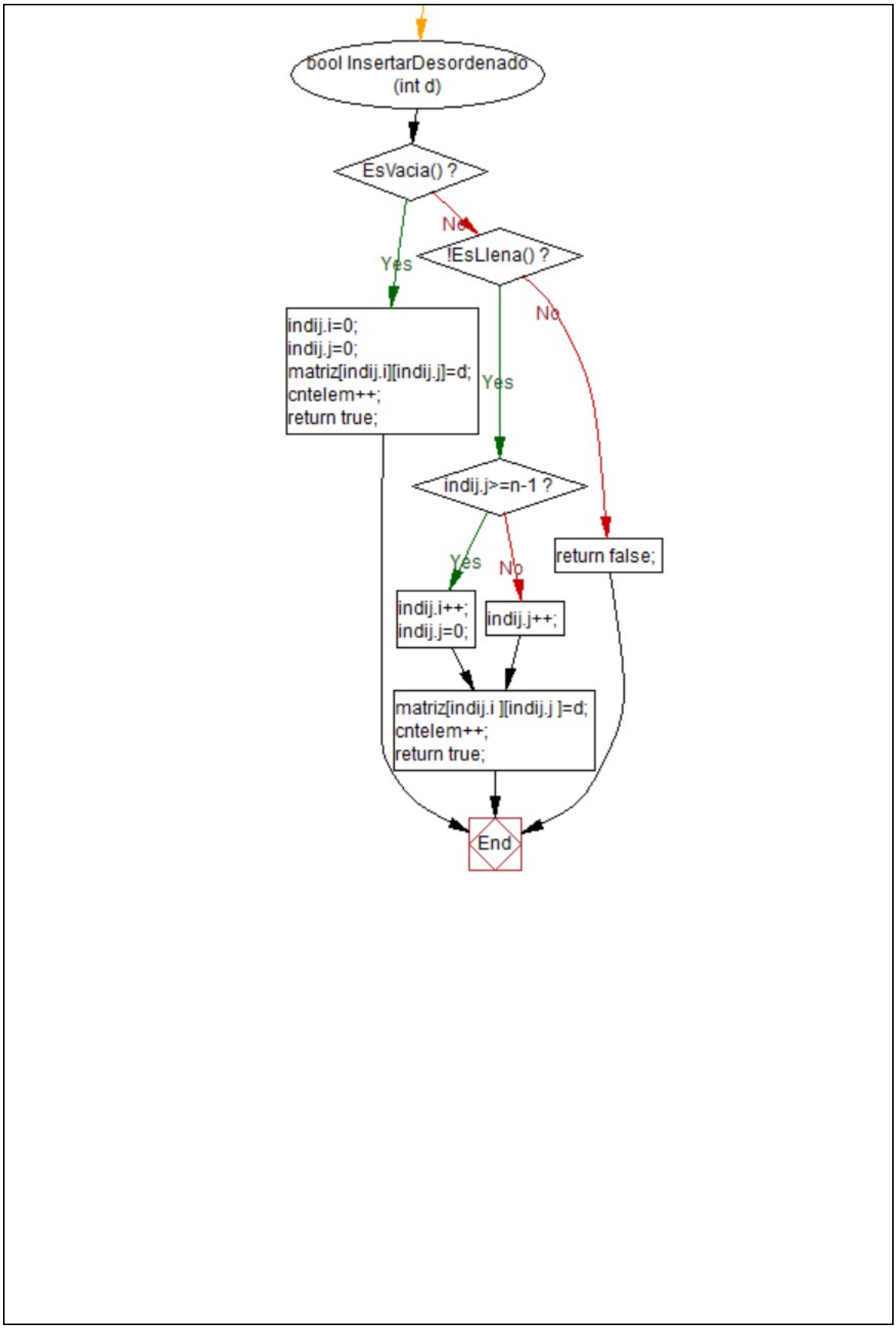
Operaciones con arreglos bidimensionales

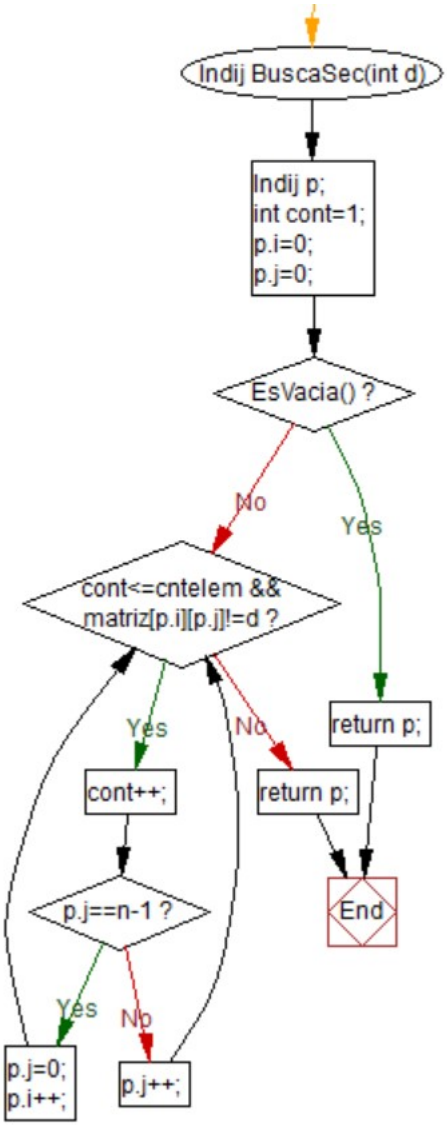
- Inserción
- Modificación
- Eliminación
- Ordenación
- Búsqueda

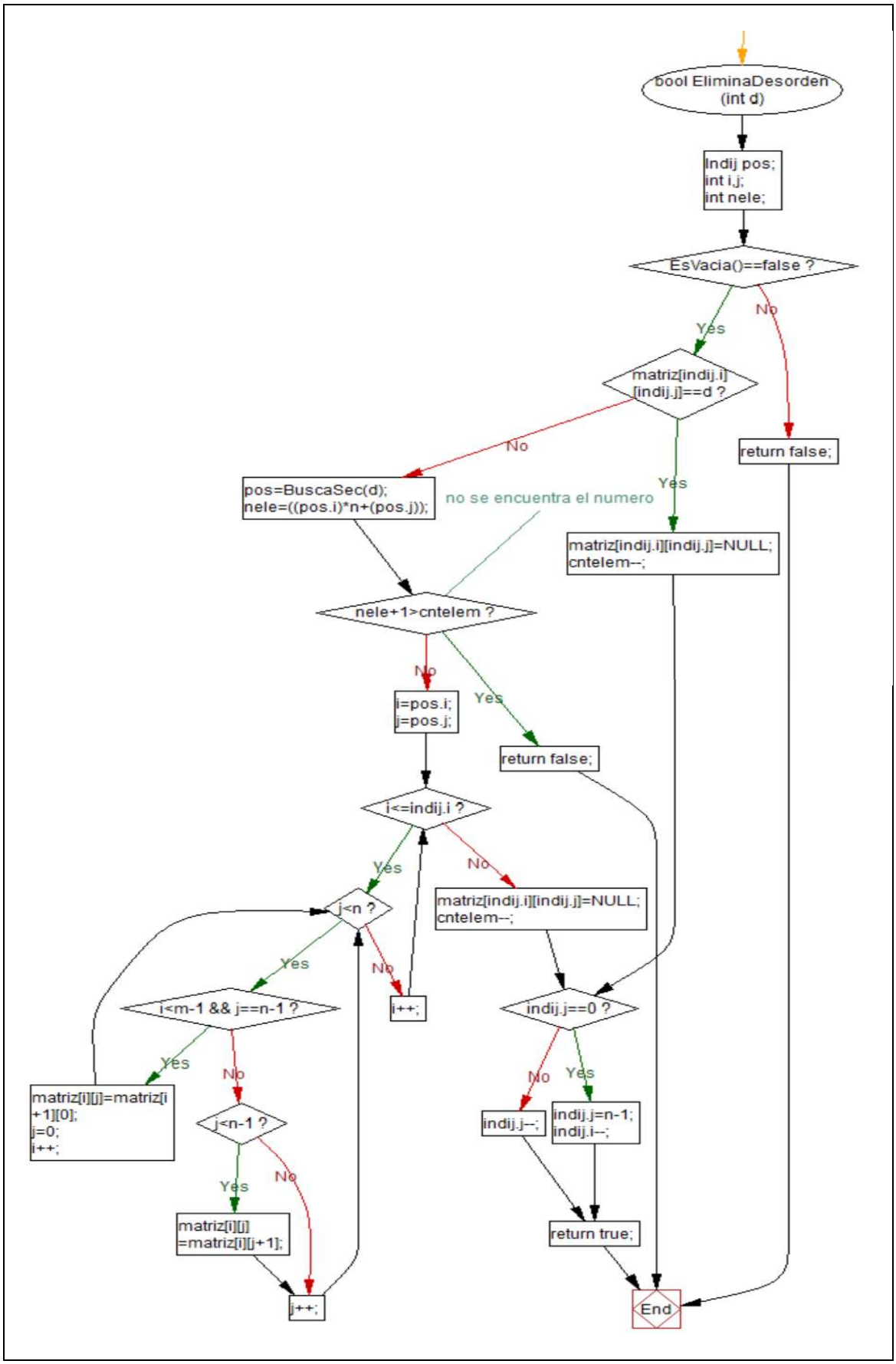
1.6.1. Algoritmo Operaciones Arreglo Bidimensional

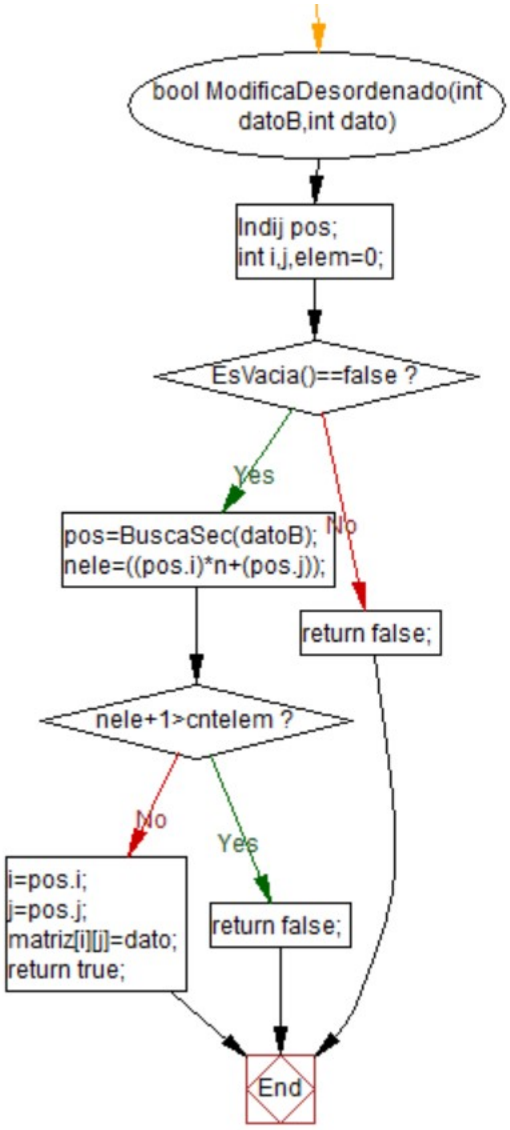
ESTRUCTURADA

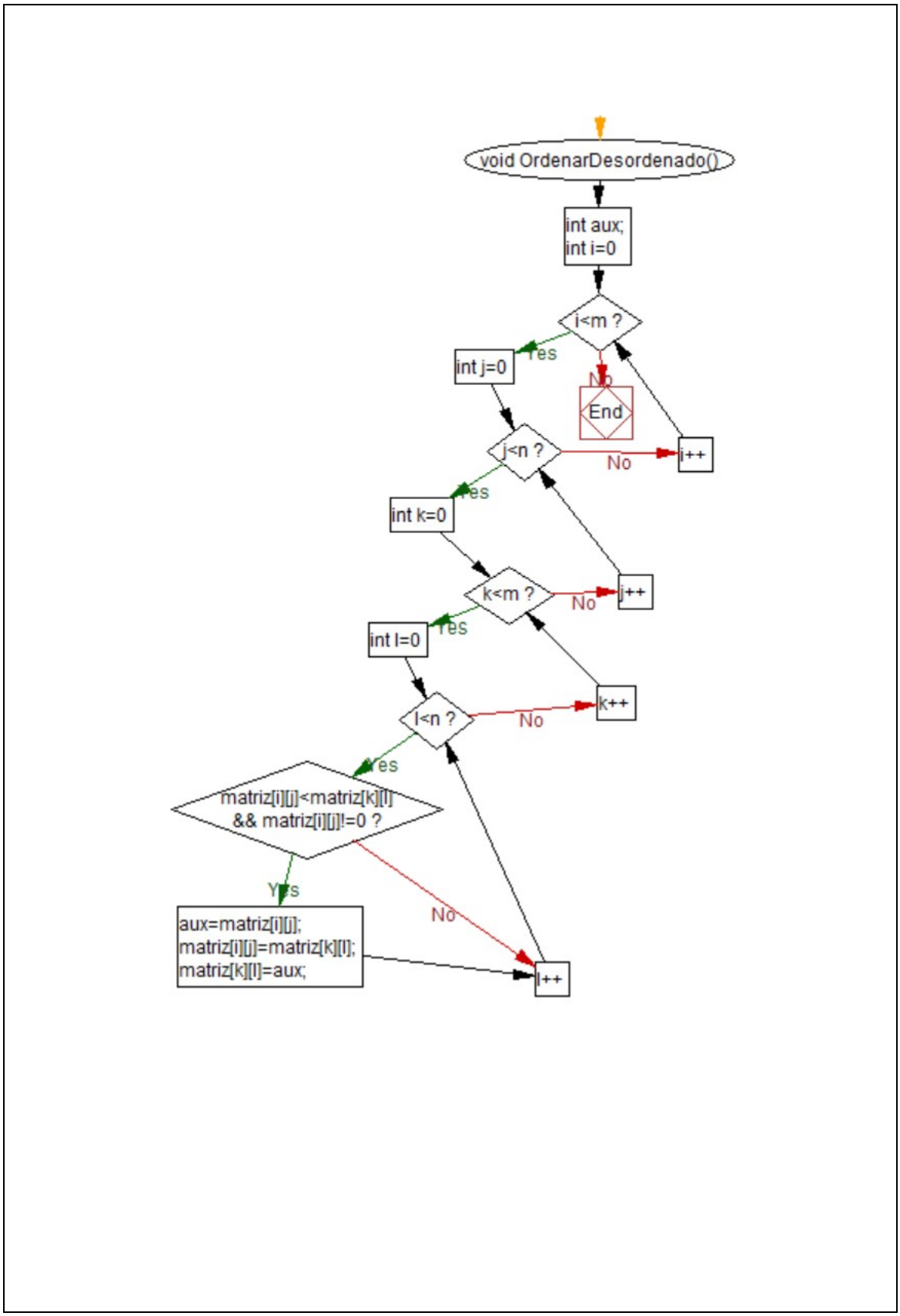

int cntelem=0,m=0,n=0;
int **matriz;
Indij indij;
int dato=0;
bool InsertarDesordenado(int);
bool EliminaDesorden(int);
void imprimir();
Indij BuscaSec(int);
bool EsVacia();
bool EsLlena();
bool ModificaDesordenado(int,int);
void OrdenarDesordenado();
End
typedef struct
int i;
int j;
End
bool EsVacia()
cntelem==0 ?
Yes
No
return true;
return false;
End
bool EsLlena()
cntelem>=(m*n) ?
Yes
No
return true;
return false;
End

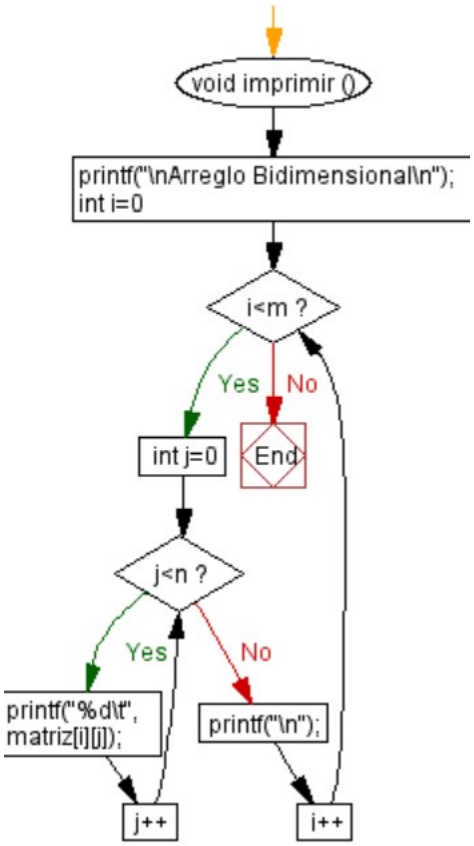

bool InsertarDesordenado
(int d)
EsVacia() ?
No
Yes
!EsLlena() ?
No
Yes
indij.i=0;
indij.j=0;
matriz[indij.i][indij.j]=d;
cntelem++;
return true;
indij.j>=n-1 ?
Yes
No
return false;
Yes
No
indij.i++;
indij.j=0;
indij.j++;
matriz[indij.i][indij.j]=d;
cntelem++;
return true;
End


Indij BuscaSec(int d)
Indij p;
int cont=1;
p.i=0;
p.j=0;
EsVacia() ?
No
Yes
cont<=cntelem &&
matriz[p.i][p.j]!=d ?
Yes
No
return p;
cont++;
return p;
p.j==n-1 ?
End
Yes
No
p.j=0;
p.i++;
p.j++;


bool EliminaDesorden (int d)
Indij pos;
int i,j;
int nele;
EsVacia()==false ?
No
Yes
return false;
matriz[indij.i] [indij.j]==d ?
No
Yes
pos=BuscaSec(d);
nele=((pos.i)*n+(pos.j));
no se encuentra el numero
matriz[indij.i][indij.j]=NULL;
cntelem--;
nele+1>cntelem ?
No
Yes
i=pos.i;
j=pos.j;
return false;
i<=indij.i ?
Yes
No
j<n ?
matriz[indij.i][indij.j]=NULL;
cntelem--;
Yes
No
i<m-1 && j==n-1 ?
i++;
indij.j==0 ?
Yes
No
No
Yes
matriz[i][j]=matriz[i +1][0];
j=0;
i++;
j<n-1 ?
indij.j--;
indij.j=n-1;
indij.i--;
Yes
No
matriz[i][j] =matriz[i][j+1];
return true;
j++;
End


bool ModificaDesordenado(int datoB,int dato)
Indij pos;
int i,j,elem=0;
EsVacia()==false ?
Yes
No
pos=BuscaSec(datoB);
nele=((pos.i)*n+(pos.j));
return false;
nele+1>cntelem ?
No
Yes
i=pos.i;
j=pos.j;
matriz[i][j]=dato;
return true;
return false;
End


void OrdenarDesordenado()
int aux;
int i=0
i<m ?
Yes
No
int j=0
End
j<n ?
No
i++
Yes
int k=0
k<m ?
No
j++
Yes
int l=0
l<n ?
No
k++
Yes
matriz[i][j]<matriz[k][l]
&& matriz[i][j]!=0 ?
Yes
No
aux=matriz[i][j];
matriz[i][j]=matriz[k][l];
matriz[k][l]=aux;
l++



Ejercicio 5.- De forma individual codificar las funciones de Arreglo Bidimensional, realice el TDAMatriz que albergue n elementos de números enteros.

*Especificación

*Niveles de abstracción

*Implementación

1.7. Arreglos tridimensionales o arreglos de arreglos

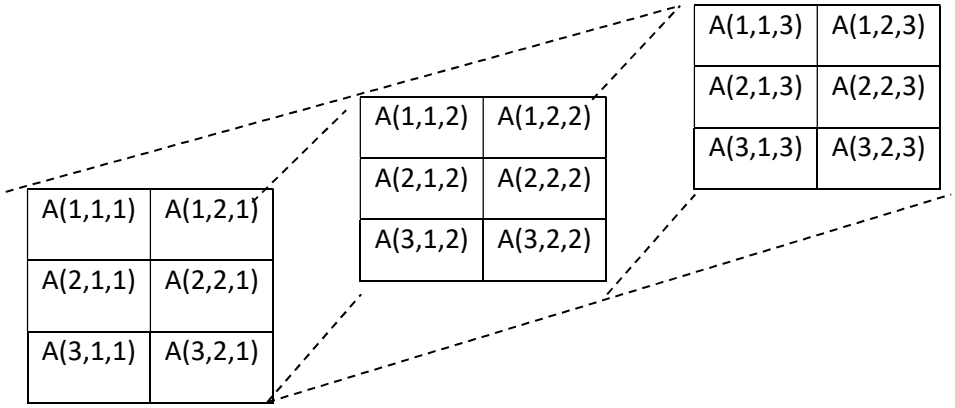
Arreglo = Array (Liminf - Limsup, Liminf - Limsup, Liminf - Limsup)

1ra dimensión 2da dimensión 3ra dimensión

Un arreglo de **n** dimensiones es una colección de uno hasta **n** elementos, para hacer a cada componente del arreglo se usarán **n** índices.

Tamaño = (Limsup$_1$ – Liminf$_1$) * (Limsup$_2$ – Liminf$_2$)* . . . . . * (Limsup$_n$ – Liminf$_n$)

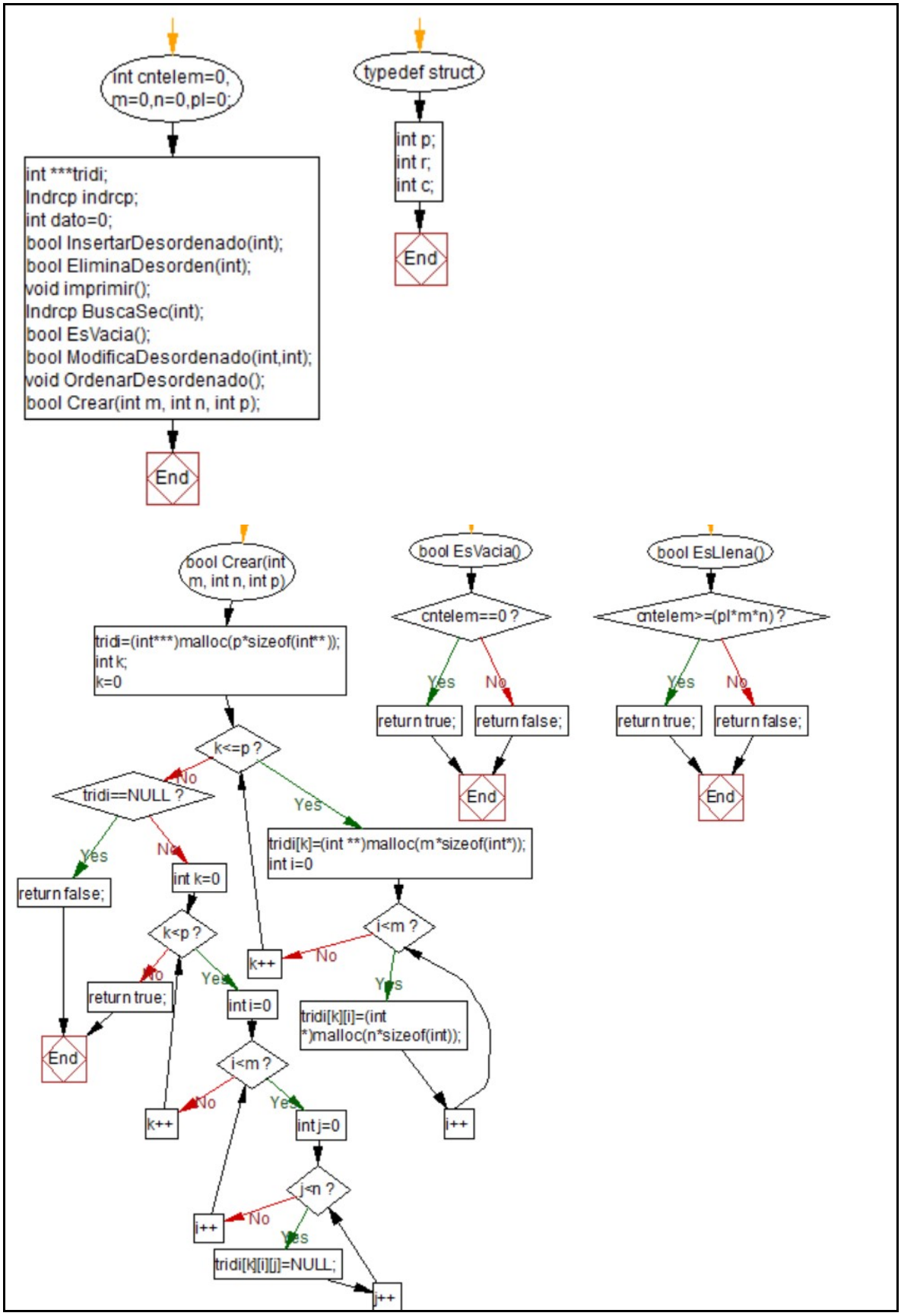
A(1.. 3, 1..2, 1..3)

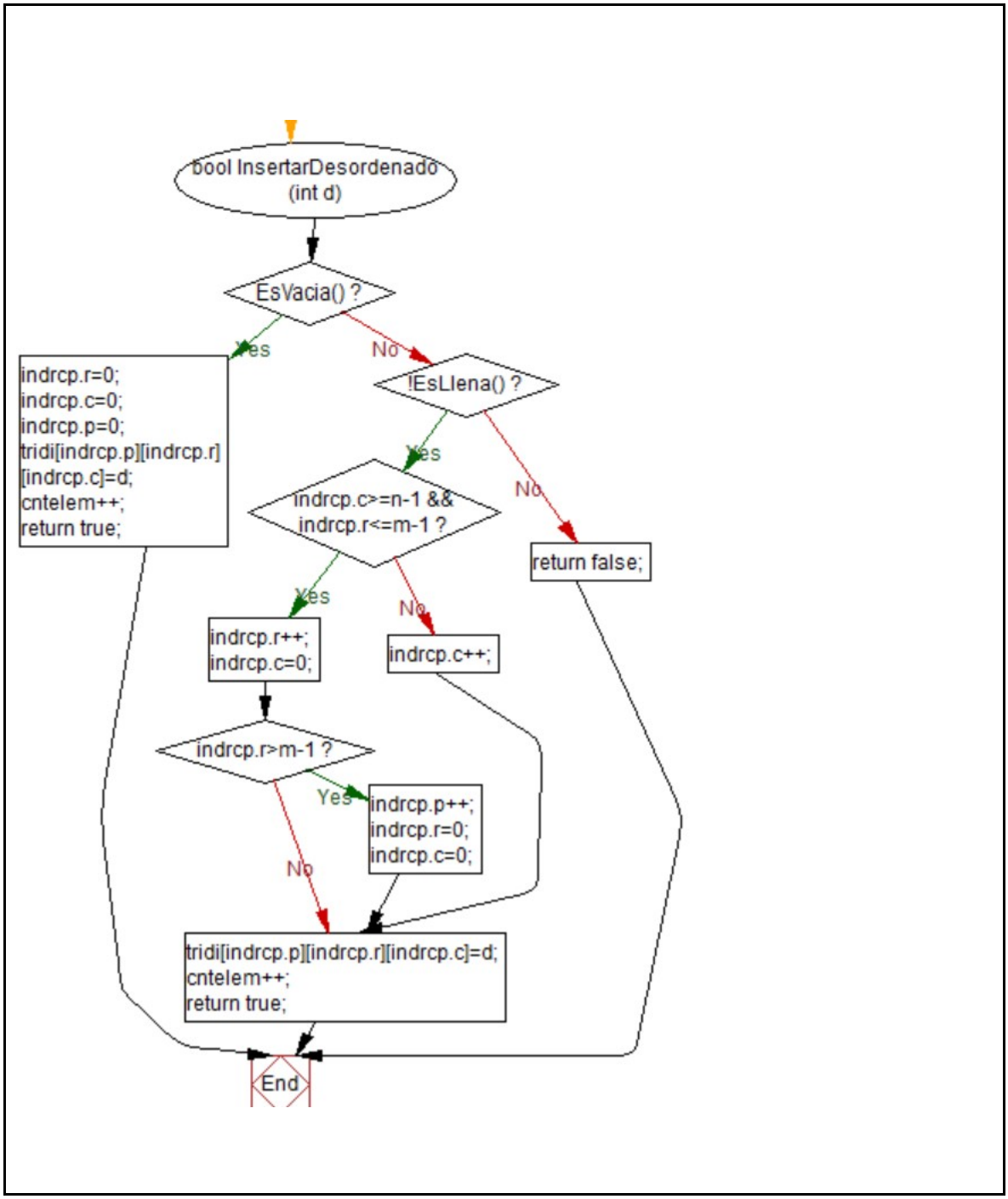


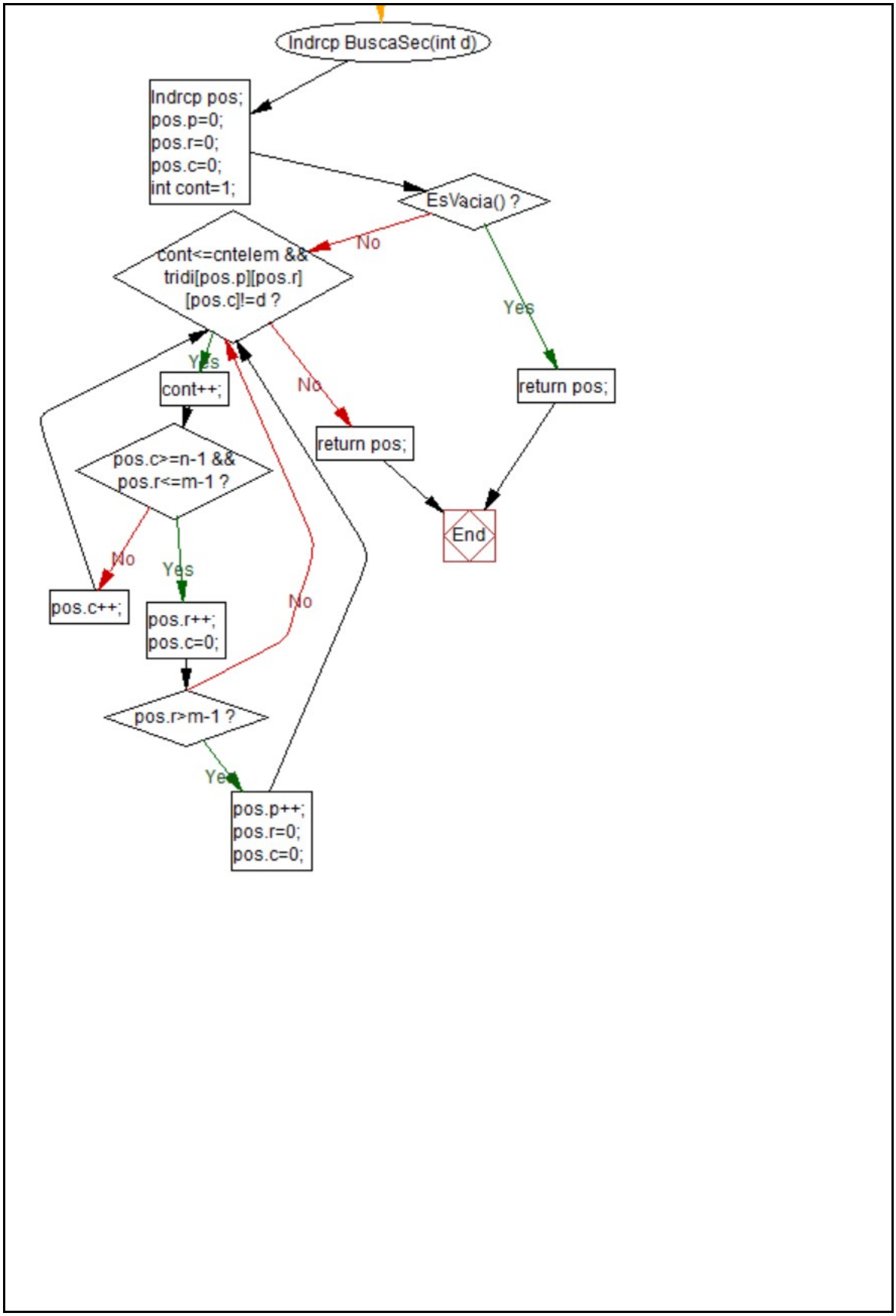
1.7.1. Operaciones

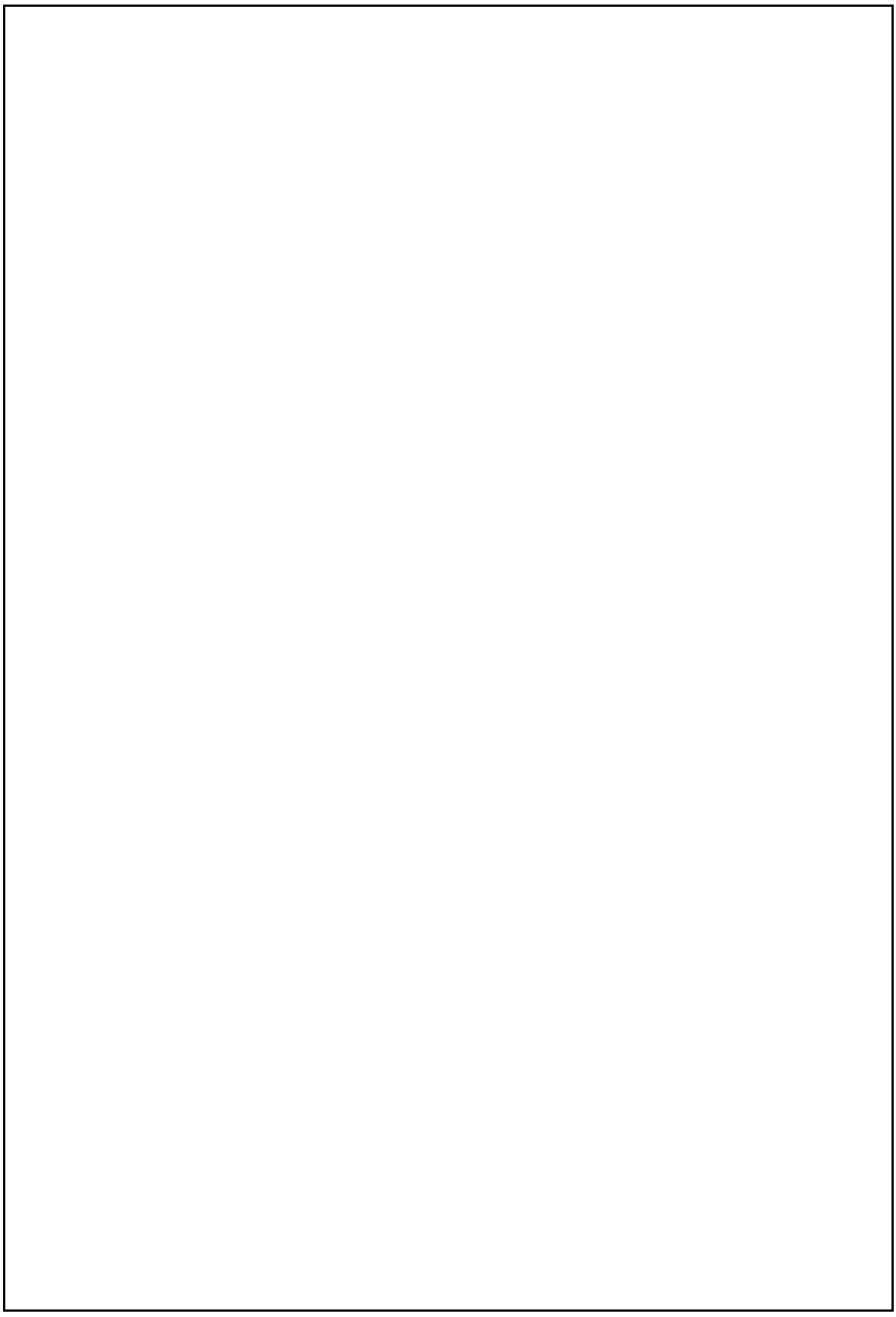
- Insertar
- Eliminar
- Modificar
- Ordenar
- Búsqueda

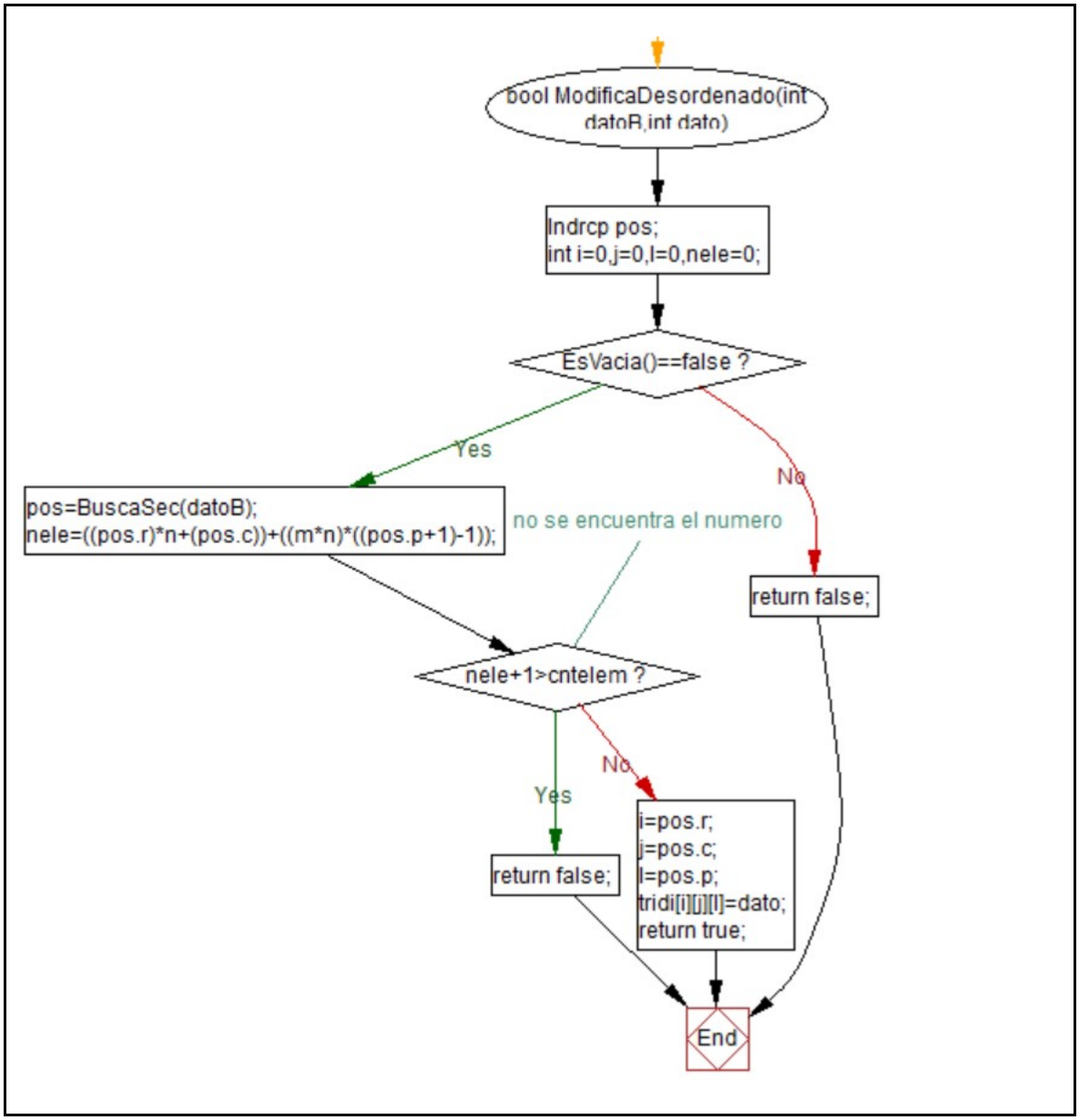
1.7.1.1. Algoritmo Operaciones Arreglo Tridimensional

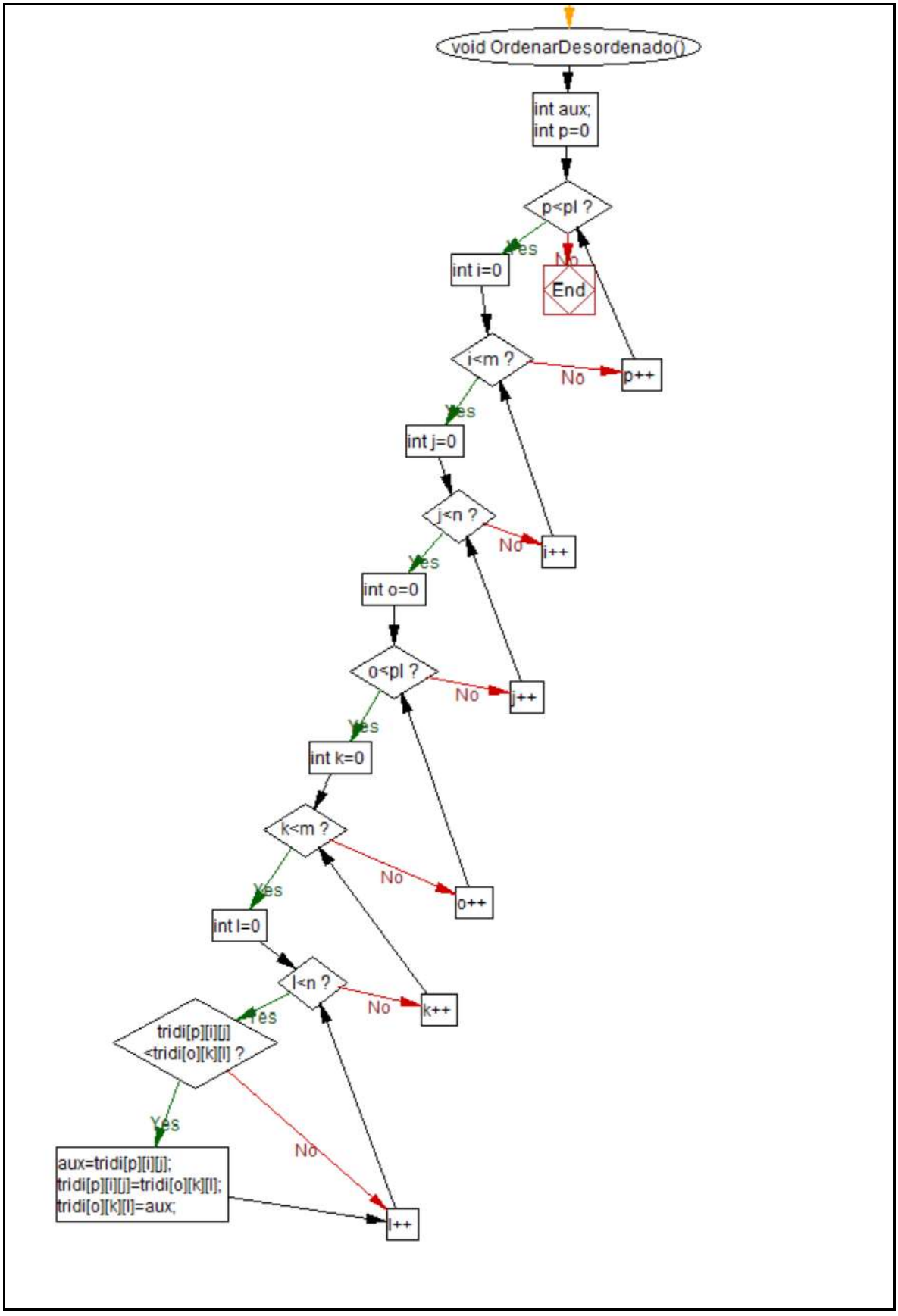

int cntelem=0, m=0,n=0,pl=0;
int ***tridi;
Indrcp indrcp;
int dato=0;
bool InsertarDesordenado(int);
bool EliminaDesorden(int);
void imprimir();
Indrcp BuscaSec(int);
bool EsVacia();
bool ModificaDesordenado(int,int);
void OrdenarDesordenado();
bool Crear(int m, int n, int p);
End
typedef struct
int p;
int r;
int c;
End
bool Crear(int m, int n, int p)
tridi=(int***)malloc(p*sizeof(int**));
int k;
k=0
k<=p ?
No
tridi==NULL ?
Yes
return false;
No
int k=0
k<p ?
No
return true;
End
Yes
int i=0
i<m ?
No
k++
Yes
tridi[k][i]=(int *)malloc(n*sizeof(int));
i++
Yes
tridi[k][i]=(int **)malloc(m*sizeof(int*));
int i=0
i<m ?
No
k++
Yes
int j=0
j<n ?
No
i++
Yes
tridi[k][i][j]=NULL;
j++
bool EsVacia()
cntelem==0 ?
Yes
return true;
No
return false;
End
bool EsLlena()
cntelem>=(pl*m*n) ?
Yes
return true;
No
return false;
End

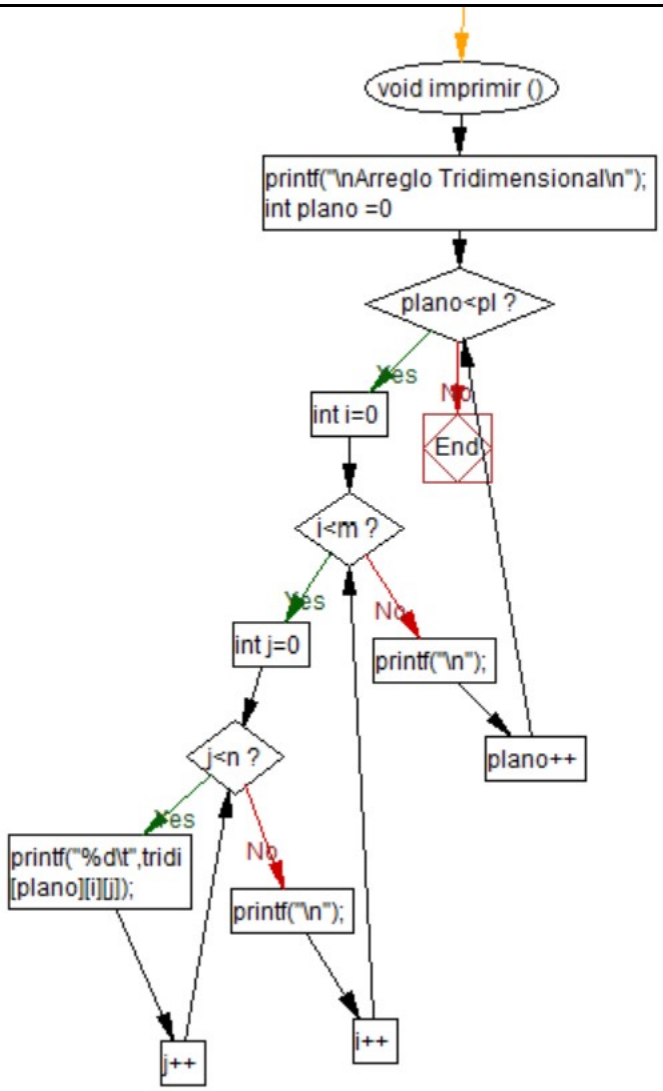

bool InsertarDesordenado
(int d)
EsVacia() ?
Yes
No
indrcp.r=0;
indrcp.c=0;
indrcp.p=0;
tridi[indrcp.p][indrcp.r]
[indrcp.c]=d;
cntelem++;
return true;
!EsLlena() ?
Yes
No
indrcp.c>=n-1 &&
indrcp.r<=m-1 ?
return false;
Yes
No
indrcp.r++;
indrcp.c=0;
indrcp.c++;
indrcp.r>m-1 ?
Yes
No
indrcp.p++;
indrcp.r=0;
indrcp.c=0;
tridi[indrcp.p][indrcp.r][indrcp.c]=d;
cntelem++;
return true;
End


Indrcp BuscaSec(int d)
Indrcp pos;
pos.p=0;
pos.r=0;
pos.c=0;
int cont=1;
EsVacia() ?
No
Yes
cont<=cntelem &&
tridi[pos.p][pos.r]
[pos.c]!=d ?
Yes
No
cont++;
return pos;
return pos;
pos.c>=n-1 &&
pos.r<=m-1 ?
End
No
Yes
pos.c++;
pos.r++;
pos.c=0;
No
pos.r>m-1 ?
Yes
pos.p++;
pos.r=0;
pos.c=0;




bool ModificaDesordenado(int datoB,int dato)
Indrcp pos;
int i=0,j=0,l=0,nele=0;
EsVacia()==false ?
Yes
No
pos=BuscaSec(datoB);
nele=((pos.r)*n+(pos.c))+((m*n)*((pos.p+1)-1));
no se encuentra el numero
return false;
nele+1>cntelem ?
No
Yes
return false;
i=pos.r;
j=pos.c;
l=pos.p;
tridi[i][j][l]=dato;
return true;
End


void OrdenarDesordenado()
int aux;
int p=0
p<pl ?
Yes
No
End
int i=0
i<m ?
No
p++
Yes
int j=0
j<n ?
No
i++
Yes
int o=0
o<pl ?
No
j++
Yes
int k=0
k<m ?
No
o++
Yes
int l=0
l<n ?
No
k++
Yes
tridi[p][i][j]
<tridi[o][k][l] ?
Yes
No
aux=tridi[p][i][j];
tridi[p][i][j]=tridi[o][k][l];
tridi[o][k][l]=aux;
l++



Ejercicio 6.- De forma individual codificar las funciones de Arreglo Bidimensional, realice el TDATridiMensional que albergue n elementos de números.

*Especificación

*Niveles de abstracción

*Implementación

Referencias

[1] R. Sedgewick and K. Wayne, Algorithms, Massachusetts: Pearson Education, Inc. , 2011.

[2] O. Cairó y S. Guardati, Estructuras de Datos, 3ra ed., D.F: McGrawHill, 2010, p. 463.

[3] M. Storti, J. D'Elía, R. Paz and L. , Algoritmos y Estructuras de Datos, Litoral, 2012.

[4] X. Franch Gutiérrez, Estructuras de Datos, especificación, diseño e implementación, 4ta ed., D.F.: Alfaomega, 2002.

[5] M. T. Goodrich y R. Tomassia, Estructuras de Datos y Algoritmos en Java, 2da ed., México: CECSA, 2002.

[6] M. A. Weiss, Estructuras de datos y algoritmos, 1995.

[7] M. A. Weiss, Estructuras de Datos en Java, Madrid: Pearson Educación S,A, 2000.

[8] N. Wirth, Algorithms and Data Structures, 1985.

[9] N. Rhodes, «data structures,» Department of Computer Science and Engineeringb, s/f. [En línea]. Available: https://datastructures201603.files.wordpress.com/2016/07/05_1_arrays_and_lists.pdf.

[10] S. Lawrence Pfleeger, Ingeniería de Software Teoría y Práctica, Brasil: Prentice Hall, 2002.