

Universidad Andres Bello  
Facultad de Ingeniería

**Proyecto**

**“Estructura NE”**

**Carrera: Ingeniería en Computación e Informática**

**Asignatura: Estructura de Datos (INS127)**

**Profesor: Tomás Lara V.**

Alumnos:

* Oclivar Regla O.
* Jorge Sepúlveda S.

Miércoles, 21 de junio de 2017

1. **DESCRIPCIÓN**

Se solicita crear un programa, en lenguaje C, que permita visualizar la imagen representada en el archivo datos.dat. Se debe considerar una matriz de 100 x 100. El archivo tiene códigos 0 (BLANCO), 1 (NEGRO), y 2 (GRIS), y el recorrido es por niveles. Esta nueva estructura se llamará NE.

Además, se solicita la implementación de las siguientes operaciones del NE sobre la estructura de datos sugerida, las cuales son:

1. La operación analizadora valorPixel considera dos casos: si la raíz tiene un color diferente a GRIS (BLANCO o NEGRO), el pixel tiene dicho valor. En caso contrario, localiza el cuadrante que contiene el pixel y desciende recursivamente.
2. La función cuadrante es una rutina de utilidad, que retorna el cuadrante en el cual el pixel [fil,col] dentro del NE. Para esto divide en dos el rango de valores en la coordenada x y el rango de valores en la coordenada y, y determina en cuál de los cuatro cuadrantes está el pixel buscado.
3. Una rutina que coloca un pixel en blanco, se considera tres casos: si la raíz es GRIS, localiza el cuadrante en el que se encuentra el pixel y hace una llamada recursiva sobre dicho subárbol. Si la dimensión del NE es 1 x 1 (solo tiene un pixel), lo coloca en BLANCO. Si la raíz del NE es de color NEGRO, la cambia por gris, crea los 4 subárboles correspondientes de color NEGRO, y finalmente hace una llamada recursiva sobre el subárbol que contiene el pixel. La rutina crear se encarga de construir un nodo de color NEGRO con las coordenadas dadas en los parámetros.
4. Una función que compare dos regiones de una imagen y encuentre que hay de común entre ellas. Esto implica la intersección de dos imágenes y determinar los puntos comunes entre estas.
5. Una función que transforme el árbol NE en un arreglo.
6. Una función que lea un arreglo y lo transforme en un árbol NE.

Por un tema de tiempo y de conocimiento, sólo se pudo implementar las operaciones 2 y 6, los cuales comentamos en el presente documento.

1. **CUERPO DEL PROYECTO**

El proyecto contempla el uso de 2 librerías más un archivo adicional que contiene la imagen en formato de números (0), (1) y (2) que pretendemos representar.

1. La primera librería llamada “Leer.h” se encarga de abrir y leer el archivo “archivo.dat”, leer el nodo raíz y los nodos hijos.

Parte de la librería dice así:

FILE\* abrirArchivo(const char\* nombre){

return fopen(nombre, "r");

}

int leerRaiz(FILE\* f){

int cero = (int) '0';

return (int)fgetc(f) - cero;

}

int\* leerHijos(FILE\* f){

int\* aux = (int\*)malloc(sizeof(int)\*4);

int i;

for(i = 0; i < 4; i++){

aux[i] = leerRaiz(f);

}

return aux;

}

1. La segunda librería llamada “Lista.h”, construye la estructura NE solicitada. Parte de la librería dice así:

typedef struct ne{

int color;

struct ne \*no;

struct ne \*ne;

struct ne \*so;

struct ne \*se;

int principiox;

int principioy;

int largo;

} NE;

typedef struct nodolista{

NE \*nodoArbol;

struct nodolista \*sgte;

struct nodolista \*ant;

}Nodolista;

typedef struct lista{

Nodolista \*ini;

Nodolista \*fin;

int tam;

}Lista;

Además, se implementaron las siguientes funciones:

1. crearLista()
2. insertarElemento()
3. recorrerLista()
4. En el fuente principal llamado “funciones.c” se definieron las funciones principales que serán utilizadas para cumplir con las operaciones antes mencionadas:
   1. cuadrante():
   2. crearArbol()
   3. buscarCuadrante()

Parte del fuente principal permite leer el tamaño de la matriz que se va a generar. Para este caso no validamos que el tamaño máximo permitido para generar la representación de la imagen fuese 100, pero de todas maneras el programa funciona en cuanto a esa definición.

A continuación de definir la dimensión de la matriz, el programa carga en memoria el contenido del archivo “archivo.dat”, luego lee posteriormente la raíz del árbol, define los cuadrantes y posteriormente crear el árbol mostrando el resultado por pantalla.

El fuente dice así:

int main (){

int tamano;

printf("------------------------------------------------\n");

printf("Ingresar la dimension de la matriz [entre 1 y 100]: ");

scanf("%d",&tamano);

printf("------------------------------------------------\n");

FILE \*f = abrirArchivo("datos.dat");

int colorRaiz = leerRaiz(f);

NE \*arbolRaiz = cuadrante(colorRaiz,0,0,tamano);

crearArbol(f,arbolRaiz,tamano);

printf("\nIngrese las coordenadas para saber en que cuadrante esta el nodo:\n ");

int x,y;

printf("\nIngrese el valor para x: ");

scanf("%i",&x);

printf("\nIngrese el valor para y: ");

scanf("%i",&y);

NE \*nodo\_buscar = cuadrante(0,x,y,0);

buscarCuadrante(arbolRaiz,nodo\_buscar);

return 0;

} //final de funcion main

Finalmente, la función permite retorna por pantalla, la representación de la imagen en números (0) y (1). Además, permite buscar el cuadrante donde se encuentra el pixel o nodo buscado.

Adjuntamos parte de la prueba realizada con su retorno.

