Universidad San Carlos de Guatemala

Facultad de Ingeniería

Estructura de Datos, Sección B

Ing. Álvaro Obrayan Hernández García

Auxiliar: Carlos Javier Castro González

Primer Semestre 2024

Tema:

Manuel Técnico – Pixel Print Studio – Proyecto Fase 2

Nombre: Carlos Manuel Lima y Lima

Registro Académico: 202201524

CUI: 3009368850101

Guatemala, 01 de abril del 2024.

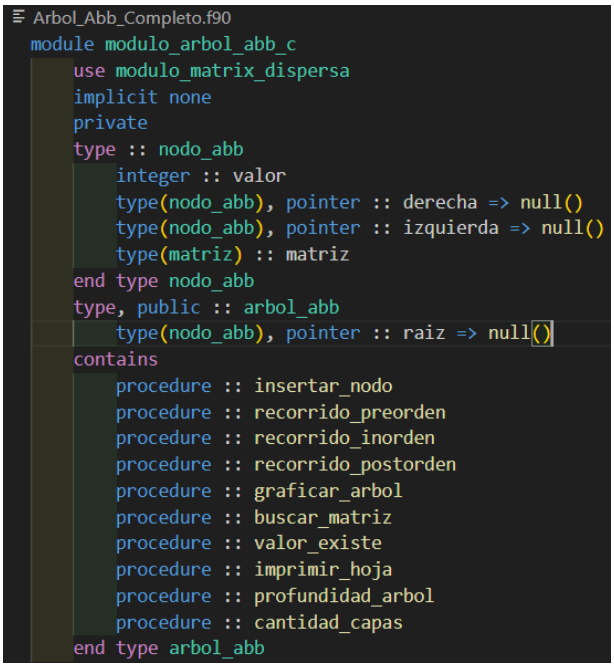
Módulo Árbol ABB Completo

Este módulo en Fortran implementa un Árbol Binario de Búsqueda (ABB). Un ABB es una estructura de datos que mantiene sus elementos en un orden específico, permitiendo operaciones eficientes de búsqueda, inserción y eliminación. Cada nodo en el árbol contiene un valor, una matriz dispersa y referencias a dos hijos: izquierdo y derecho. El código define un módulo llamado modulo_arbol_abb_c que contiene la definición de los tipos nodo_abb y arbol_abb, y una serie de subrutinas y funciones para manipular estos tipos.

Sus principales subrutinas son:

- ***insertar_nodo:*** Inserta un nuevo nodo en el árbol. Si el árbol está vacío, el nuevo nodo se convierte en la raíz. Si no, el nuevo nodo se inserta en el lugar correcto en el árbol según las reglas del ABB.
- ***recorrido_preorden:*** Realiza un recorrido en preorden del árbol. En un recorrido en preorden, se visita primero la raíz, luego el subárbol izquierdo y finalmente el subárbol derecho.
- ***recorrido_inorden:*** Realiza un recorrido en inorden del árbol. En un recorrido en inorden, se visita primero el subárbol izquierdo, luego la raíz y finalmente el subárbol derecho.
- ***recorrido_postorden:*** Realiza un recorrido en postorden del árbol. En un recorrido en postorden, se visita primero el subárbol izquierdo, luego el subárbol derecho y finalmente la raíz.
- ***graficar_arbol:*** Genera un gráfico del árbol utilizando el formato DOT para graficar el árbol.
- ***buscar_matriz:*** Devuelve la matriz dispersa asociada a un valor dado en el árbol.
- ***valor_existe:*** Verifica si un valor dado existe en el árbol.
- ***imprimir_hoja:*** Imprime las hojas del árbol.
- ***profundidad_arbol:*** Calcula la profundidad del árbol.
- ***cantidad_capas:*** Calcula la cantidad de capas en el árbol.

Cada una de estas subrutinas y funciones juega un papel crucial en la manipulación del árbol binario de búsqueda. Juntas, proporcionan una implementación completa y robusta de un ABB en Fortran.


Arbol_Abb_Completo.f90

```
module modulo_arbol_abb_c
    use modulo_matrix_dispersa
    implicit none
    private
    type :: nodo_abb
        integer :: valor
        type(nodo_abb), pointer :: derecha => null()
        type(nodo_abb), pointer :: izquierda => null()
        type(matriz) :: matriz
    end type nodo_abb
    type, public :: arbol_abb
        type(nodo_abb), pointer :: raiz => null()
    contains
        procedure :: insertar_nodo
        procedure :: recorrido_preorden
        procedure :: recorrido_inorden
        procedure :: recorrido_postorden
        procedure :: graficar_arbol
        procedure :: buscar_matriz
        procedure :: valor_existe
        procedure :: imprimir_hoja
        procedure :: profundidad_arbol
        procedure :: cantidad_capas
    end type arbol_abb
```

Modulo Árbol ABB Simple

Este código en Fortran implementa un Árbol Binario de Búsqueda (ABB) de una manera más simple que el código anterior. Un ABB es una estructura de datos que mantiene sus elementos en un orden específico, permitiendo operaciones eficientes de búsqueda, inserción y eliminación. Cada nodo en el árbol contiene un valor y referencias a dos hijos: izquierdo y derecho.

El código define un módulo llamado modulo_arbol_abb_s que contiene la definición de los tipos nodo_abb_simple y arbol_abb_simple, y una serie de subrutinas y funciones para manipular estos tipos.

Sus principales subrutinas son:

insertar: Esta subrutina inserta un nuevo nodo en el árbol. Si el árbol está vacío, el nuevo nodo se convierte en la raíz. Si no, el nuevo nodo se inserta en el lugar correcto en el árbol según las reglas del ABB.

recorrido_amplitud: Realiza un recorrido en amplitud (o por niveles) del árbol. En un recorrido en amplitud, se visitan todos los nodos de un nivel antes de pasar al siguiente.

generar_grafica: Esta subrutina genera un gráfico del árbol. Para hacer esto, recorre el árbol y genera cadenas de texto para crear nodos y enlaces entre ellos, que se utilizarán para graficar el árbol.

numero_nodos: Esta función calcula el número total de nodos en el árbol.

Estas subrutinas y funciones proporcionan una implementación completa y robusta de un ABB en Fortran. Juntas, permiten la manipulación y visualización del árbol binario de búsqueda.

Arbol_Abb_Simple.f90

```
module modulo_arbol_abb_s
    implicit none
    private
    type :: nodo_abb_simple
        integer :: valor
        type(nodo_abb_simple), pointer :: derecha => null()
        type(nodo_abb_simple), pointer :: izquierda => null()
    end type nodo_abb_simple
    type, public :: arbol_abb_simple
        type(nodo_abb_simple), pointer :: raiz => null()
    contains
        procedure :: insertar
        procedure :: recorrido_amplitud
        procedure :: generar_grafica
        procedure :: numero_nodos
    end type arbol_abb_simple

contains
```

Modulo Árbol AVL Completo

Este código en Fortran define un módulo llamado modulo_arbol_avl_c que implementa un árbol AVL (Adelson-Velsky y Landis). Un árbol AVL es un tipo especial de árbol binario de búsqueda autoequilibrado que mantiene su altura equilibrada para garantizar operaciones eficientes de búsqueda, inserción y eliminación.

Cada nodo en el árbol AVL de este código Fortran contiene un valor entero, una altura que representa la altura del nodo en el árbol, punteros a los nodos hijos derecho e izquierdo, y un árbol binario de búsqueda simple interno. Estos componentes permiten que el árbol AVL realice operaciones de búsqueda, inserción y eliminación de manera eficiente, manteniendo al mismo tiempo el equilibrio del árbol.

Sus principales subrutinas son:

- ***insertar_nodo:*** Esta subrutina inserta un nuevo nodo en el árbol AVL. Si el árbol está vacío, se crea un nuevo nodo y se establece como la raíz. Si el árbol no está vacío, la subrutina busca el lugar correcto para insertar el nuevo nodo y luego realiza las rotaciones necesarias para mantener el árbol equilibrado.
- ***eliminar_nodo:*** Esta subrutina elimina un nodo del árbol AVL. Busca el nodo con el valor dado, lo elimina y luego realiza las rotaciones necesarias para mantener el árbol equilibrado.
- ***buscar_valor:*** Esta subrutina busca un valor en el árbol AVL. Si el valor está presente en el árbol, devuelve el nodo que contiene ese valor.
- ***valor_existe:*** Esta subrutina verifica si un valor existe en el árbol AVL. Devuelve un valor lógico que indica si el valor está presente en el árbol.
- ***top_5_imagenes:*** Esta subrutina recorre el árbol AVL y encuentra las cinco imágenes con la mayor cantidad de capas. Imprime los ID de las imágenes y la cantidad total de capas.
- ***graficar_arbol:*** Esta subrutina genera una representación gráfica del árbol AVL utilizando el software Graphviz. Crea un archivo .dot con la estructura del árbol y luego lo convierte en un archivo .pdf.
- ***graficar_arbol_imagen:*** Similar a graficar_arbol, pero también destaca un valor específico en el árbol AVL y genera su árbol ABB asociado.
- ***cantidad_imagenes:*** Esta subrutina cuenta la cantidad total de imágenes en el árbol AVL y la imprime.

Arbol_Avl_Completo.f90

```
module modulo_arbol_avl_c
    use modulo_arbol_abb_s
    implicit none
    private
    type :: nodo_avl
        integer :: valor
        integer :: altura = 1
        type(nodo_avl), pointer :: derecha => null()
        type(nodo_avl), pointer :: izquierda => null()
        type(arbol_abb_simple) :: arbol_interno
    end type
```

```
type, public :: arbol_avl
    type(nodo_avl), pointer :: raiz => null()
contains
    procedure :: insertar_nodo
    procedure :: eliminar_nodo
    procedure :: buscar_valor
    procedure :: valor_existe
    procedure :: top_5_imagenes
    procedure :: graficar_arbol
    procedure :: graficar_arbol_imagen
    procedure :: cantidad_imagenes
end type arbol_avl
```

Modulo Lista Imagen

Este código en Fortran define un módulo llamado modulo_lista_imagen que implementa una lista enlazada de imágenes. Cada nodo en la lista contiene una imagen y un puntero al siguiente nodo.

nodo_imagen: Este es el tipo de datos para los nodos en la lista enlazada. Cada nodo_imagen contiene una imagen (una cadena de caracteres) y un puntero al siguiente nodo_imagen en la lista.

lista_imagen: Este es el tipo de datos para la lista enlazada en sí. Contiene un puntero a la cabeza de la lista y una serie de procedimientos para manipular la lista.

Sus principales subrutinas son:

insertar_imagen: Esta subrutina inserta una nueva imagen al principio de la lista. Crea un nuevo nodo_imagen, asigna la imagen al nodo, y luego ajusta los punteros para que el nuevo nodo sea ahora la cabeza de la lista.

Este módulo proporciona una forma eficiente de manejar una colección de imágenes, permitiendo la inserción rápida de nuevas imágenes al principio de la lista. Sin embargo, como con cualquier implementación de una estructura de datos, es importante tener en cuenta las características específicas del conjunto de datos y las operaciones requeridas para determinar si una lista enlazada es la estructura de datos más adecuada para el problema en cuestión.

Lista_Imagen_Cliente.f90

```
module modulo_lista_imagen
    implicit none
    type :: nodo_imagen
        character(len=:), allocatable :: imagen
        type(nodo_imagen), pointer :: siguiente => null()
    end type nodo_imagen
    type :: lista_imagen
        type(nodo_imagen), pointer :: cabeza => null()
        contains
        procedure :: insertar_imagen
    end type lista_imagen

    contains
    subroutine insertar_imagen(self, imagen)
        class(lista_imagen), intent(inout) :: self
        character(len=*), intent(in) :: imagen
        type(nodo_imagen), pointer :: nuevo_nodo
        allocate(nuevo_nodo)
        nuevo_nodo%imagen = imagen
        nuevo_nodo%siguiente => self%cabeza
        self%cabeza => nuevo_nodo
    end subroutine insertar_imagen

end module modulo_lista_imagen
```

Modulo Lista Álbumes

Este código en Fortran define un módulo llamado md que contiene una estructura de datos para una lista doblemente enlazada de álbumes, donde cada álbum contiene una lista de imágenes.

Sus principales subrutinas son:

insertar_album: Esta subrutina toma un nombre de álbum y una lista de imágenes, crea un nuevo nodo de álbum, y lo inserta en la lista de álbumes. Si la lista está vacía, el nuevo nodo se convierte en la cabeza de la lista. Si no, se inserta al final de la lista.

graficar_album: Esta subrutina genera un gráfico de la lista de álbumes utilizando el software Graphviz. Cada nodo del gráfico representa un álbum y sus imágenes asociadas. El gráfico se guarda en un archivo PDF.

eliminar_imagen: Esta subrutina recorre todos los álbumes en la lista y elimina todas las instancias de una imagen específica. Si la imagen a eliminar es la primera en la lista de imágenes de un álbum, se actualiza la cabeza de la lista.

imprimir_albumes: Esta subrutina imprime todos los álbumes y sus imágenes asociadas. Si la lista de álbumes está vacía, imprime un mensaje indicando que no álbumes para mostrar.

En conclusión, este código proporciona una implementación robusta y eficiente para manejar una colección de álbumes y sus imágenes asociadas. Las operaciones de inserción, eliminación, impresión y graficación están bien definidas y manejan adecuadamente los casos de borde. Este módulo podría ser una excelente base para desarrollar una aplicación de gestión de álbumes de fotos.

Lista_Albumes_Cliente.f90

```
module md
    use modulo_lista_imagen
    implicit none
    type :: nodo_album
        character(len=:), allocatable :: album
        type(lista_imagen) :: lista_imagenes
        type(nodo_album), pointer :: anterior, siguiente
    end type nodo_album
    type :: lista_album
        type(nodo_album), pointer :: cabeza => null()
        contains
        procedure :: insertar_album
        procedure :: graficar_album
        procedure :: eliminar_imagen
        procedure :: imprimir_albumes
    end type lista_album
```

Modulo Lista Clientes

Este código en Fortran define un módulo llamado modulo_lista_cliente que contiene una estructura de datos para una lista enlazada de clientes. Cada cliente tiene un DPI, un nombre, una contraseña, un árbol ABB de capas, un árbol AVL de imágenes y una lista doblemente enlazada de álbumes.

nodo_cliente: Este es un tipo definido por el usuario que representa un nodo en la lista de clientes. Cada nodo tiene un DPI, un nombre, una contraseña, un árbol ABB de capas, un árbol AVL de imágenes, una lista doblemente enlazada de álbumes y un puntero al siguiente nodo en la lista.

Sus principales subrutinas son:

- ***insertar_cliente:*** Esta subrutina toma un DPI, un nombre y una contraseña, crea un nuevo nodo de cliente, y lo inserta al principio de la lista de clientes.
- ***eliminar_cliente:*** Esta subrutina elimina un cliente de la lista utilizando su DPI. Si el cliente no existe, imprime un mensaje indicando que el cliente no está registrado. Si el cliente existe, se elimina de la lista y se libera la memoria asociada.
- ***modificar_cliente:*** Esta subrutina modifica la información de un cliente existente en la lista utilizando su DPI. Si el cliente no existe, imprime un mensaje indicando que el cliente no está registrado.
- ***grafica_cliente:*** Esta subrutina genera un gráfico de la lista de clientes utilizando el software Graphviz. Cada nodo del gráfico representa un cliente y sus detalles. El gráfico se guarda en un archivo PDF.
- ***cliente_existe:*** Esta función verifica si un cliente existe en la lista utilizando su DPI.
- ***iniciar_sesion_c:*** Esta función verifica si las credenciales de un cliente son correctas.
- ***obtener_cliente:*** Esta función devuelve un puntero al nodo de un cliente específico en la lista utilizando su DPI.
- ***mostrar_cliente:*** Esta subrutina imprime la información de un cliente específico.
- ***reporte_albumes_cliente:*** Esta subrutina imprime todos los álbumes asociados a un cliente específico.
- ***reporte_imagenes_cliente:*** Esta subrutina imprime todas las imágenes asociadas a un cliente específico utilizando el árbol AVL de imágenes.
- ***reporte_capas_cliente:*** Esta subrutina imprime todas las capas asociadas a un cliente específico utilizando el árbol ABB de capas.
- ***reporte_listar_cliente:*** Esta subrutina imprime todos los clientes registrados y sus imágenes asociadas.

Este código proporciona una implementación robusta y eficiente para manejar una colección de clientes y sus álbumes e imágenes asociadas. Las operaciones de inserción, eliminación, modificación, impresión y generación de informes están bien definidas y manejan adecuadamente los casos de borde. Este módulo podría ser una excelente base para desarrollar una aplicación de gestión de clientes y sus álbumes de fotos. Además, la capacidad de generar gráficos de la lista de clientes puede ser útil para visualizar la estructura de la lista y los detalles de los clientes.

Modulo Matriz Dispersa

Este código en Fortran define un módulo llamado modulo_matrix_dispersa que contiene una estructura de datos para una matriz dispersa. Cada nodo en la matriz representa un píxel con un color específico.

Contenido de cada nodo:

- ***nodo_valor:*** Este es un tipo definido por el usuario que representa un valor en la matriz. Cada nodo tiene un color y un indicador de existencia.
- ***nodo_pixel:*** Este es otro tipo definido por el usuario que representa un píxel en la matriz. Cada nodo tiene una fila, una columna, un color y punteros a los nodos arriba, abajo, derecha e izquierda.

Principales subrutinas:

- ***insertar_nodo:*** Esta subrutina inserta un nuevo nodo en la matriz en la posición especificada por las coordenadas de fila y columna. Si la matriz está vacía, se crea una raíz.
- ***insertar_matriz:*** Esta subrutina inserta una matriz completa en la matriz actual.
- ***insertar_cabecera_columna:*** Esta subrutina inserta una nueva cabecera de columna en la matriz.
- ***insertar_cabecera_fila:*** Esta subrutina inserta una nueva cabecera de fila en la matriz.
- ***insertarEnFila:*** Esta subrutina inserta un nuevo nodo en la fila correcta de la matriz dispersa.
- ***insertarEnColuma:*** Esta subrutina inserta un nuevo nodo en la columna correcta de la matriz dispersa.
- ***buscar_fila:*** Esta función devuelve un puntero al nodo de la fila especificada en la matriz dispersa.
- ***buscar_columna:*** Esta función devuelve un puntero al nodo de la columna especificada en la matriz dispersa.
- ***existe_nodo:*** Esta función verifica si un nodo existe en la matriz.
- ***graficar_matriz:*** Esta subrutina genera un gráfico de la matriz dispersa utilizando el software Graphviz. Cada nodo del gráfico representa un píxel y su color. El gráfico se guarda en un archivo PNG.

Este código proporciona una implementación robusta y eficiente para manejar una matriz dispersa, que es una estructura de datos útil para representar una matriz donde la mayoría de los elementos son cero. Las operaciones de inserción, búsqueda y graficación están bien definidas y manejan adecuadamente los casos de borde. Este módulo podría ser una excelente base para desarrollar una aplicación que requiera el uso de matrices dispersas, como el procesamiento de imágenes. Además, la capacidad de generar gráficos de la matriz puede ser útil para visualizar la estructura de la matriz y los colores de los píxeles.

Main

Este código en Fortran define un programa principal que gestiona una aplicación de manipulación de imágenes. El programa utiliza varios módulos para manejar diferentes aspectos de la aplicación, incluyendo la manipulación de archivos JSON, la gestión de listas de clientes e imágenes, y la manipulación de árboles ABB y AVL.

El programa proporciona una interfaz de usuario que permite a los usuarios iniciar sesión o registrarse, y ofrece diferentes menús para los administradores y los clientes. Los administradores pueden gestionar los clientes y generar varios informes, mientras que los clientes pueden gestionar sus álbumes e imágenes.

Las principales subrutinas del programa son:

- ***mostrar_menu:*** Muestra el menú principal de la aplicación.
- ***iniciar_sesion:*** Permite a los usuarios iniciar sesión en la aplicación.
- registrarse: Permite a los nuevos usuarios registrarse en la aplicación.
- ***menu_administrador:*** Muestra el menú del administrador, que permite al administrador gestionar los clientes y generar informes.
- ***abc_cliente:*** Muestra el menú de gestión de clientes, que permite al administrador agregar, eliminar o modificar clientes.
- ***reportes_administrador:*** Muestra el menú de informes del administrador.
- ***menu_cliente:*** Muestra el menú del cliente, que permite al cliente visualizar estructuras, navegar y gestionar imágenes, realizar una carga masiva de información, generar informes y manejar imágenes.
- ***visualizar_estructura:*** Muestra el menú de visualización de estructuras, que permite al cliente ver el árbol de capas, el árbol de imágenes, la lista de álbumes, una capa específica, una imagen específica y su árbol de capas.
- ***generador_imagen:*** Muestra el menú del generador de imágenes, que permite al cliente generar una imagen por recorrido limitado, por árbol de imágenes, o por capas.
- ***carga_masiva:*** Muestra el menú de carga masiva, que permite al cliente cargar capas, imágenes o álbumes de forma masiva.
- ***reportes_usuario:*** Muestra el menú de informes del usuario.
- ***abc_imagen:*** Muestra el menú de gestión de imágenes, que permite al cliente registrar una nueva imagen, eliminar una imagen existente o regresar al menú del cliente.
- ***carga_masiva_capa:*** Permite al cliente cargar capas de forma masiva desde un archivo JSON.
- ***carga_masiva_imagen:*** Permite al cliente cargar imágenes de forma desde un archivo JSON.
- ***carga_masiva_album:*** Permite al cliente cargar álbumes de forma masiva desde un archivo JSON.
- ***carga_masiva_cliente:*** Permite al cliente cargar clientes de forma masiva desde un archivo JSON.

Este programa demuestra un buen uso de las estructuras de datos y las capacidades de programación en Fortran para desarrollar una aplicación interactiva y fácil de usar. Además, la capacidad de generar gráficos de la matriz puede ser útil para visualizar la estructura de la matriz y los colores de los píxeles.