Universidad de Costa Rica

Facultad de Ingeniería

Escuela de Computación e Informática

Estructuras de Datos y Análisis de Algoritmos

CI-1221

Grupo 04

II Etapa I TP

Tarea Programada 1 Análisis de algoritmos y estructuras de datos.

Profesora:

Sandra Kikut

Elaborado por:

Carrión Claeys Archibald C01736

Fernando Arce Castillo C10572

Saltachín Solano Javier C17632

Día 19 de mes 10 del año 2022

Tabla de Contenidos Mínimos

Objetivo general:

Comprender el funcionamiento e implementación de los modelos lógicos cola, lista y arbol n-ario.

Objetivos específicos:

Implementar el modelo lógico cola, mediante la estructura de datos arreglo circular.

Implementar el modelo lógico lista indexada, mediante la estructura de datos lista simplemente enlazada.

Implementar el modelo lógico arbol n-ario, mediante las estructuras de datos arreglo con señalador al padre, lista de hijos e Hijo Más Izquierdo-Hermano Derecho.

Modelo Cola.

Cola: El modelo lógico Cola consiste en un grupo de datos, tal que ellos se encuentran ordenados linealmente (sin ramificaciones), de manera análoga a una fila de espera. En él solo se pueden insertar datos por un extremo, y extraer datos por el otro extremo.

Esta inserción-extracción corresponde al orden FIFO (First in First Out), es decir, que el Elemento que entró primero respecto a los demás es el primero en salir del modelo.

Definición y especificación de operadores básicos del modelo Cola.

• Crear

Nombre	Crear
Parámetros	Una Cola
Efecto	Inicializa una estructura de datos para representar al modelo lógico Cola
Requiere	Una Cola que no esté inicializada
Modifica	Crea una estructura de datos

• Destruir

Nombre	Destruir
Parámetros	Una Cola inicializada aún no destruida

Efecto	Borra la estructura Cola y los datos que contiene
Requiere	Una Cola inicializada
Modifica	Elimina la estructura de datos, libera el espacio utilizado por la estructura

• Encolar

Nombre	Encolar
Parámetros	Una Cola, un elemento
Efecto	Agrega al inicio de la Cola el elemento dado por parámetro
Requiere	Una Cola inicializada, un Elemento válido
Modifica	La cantidad total de datos en la Cola, agrega un elemento a la Cola

• Desencolar?

Nombre	Desencolar
Parámetros	Una Cola
Efecto	Devuelve un elemento almacenado en la Cola (por definición, el primero que haya entrado)
Requiere	Una Cola inicializada
Modifica	El tamaño de la Cola disminuye en una unidad y se extrae un elemento.

• NumElem

Nombre	NumElem
Parámetros	Una Cola
Efecto	Devuelve un valor entero igual a la cantidad de elementos en la Cola
Requiere	Una Cola inicializada
Modifica	No modifica la estructura de datos.

Modelo Lista Indexada

Lista Indexada: Es un modelo lógico que extiende a manera de especialización a la Lista Posicionada. Consiste en un contenedor de agrupación concreta, finita, ordenada y discreta de elementos (agnósticos al tipo de dato agrupado), donde se dice que cada elemento tiene una posición ordinal en la agrupación.

Las posiciones de un elemento en una Lista Indexada tienen un índice numérico discreto concreto conocido. Para esta Lista se puede hablar de la i-ésima posición de elementos, y es consenso que el índice de aquella posición esté contenido en un rango acotado por el tamaño de la Lista.

Definición y especificación de operadores básicos del modelo Cola.

• Crear

Nombre	Crear
Parámetros	Una Lista Indexada
Efecto	Inicializa una estructura de datos subyacente para representar al modelo.
Requiere	Que la Lista Indexada alimentada como parámetro esté sin inicializar
Modifica	La Lista Indexada tiene su estructura de datos subyacente inicializada.

• Destruir

Nombre	Destruir
Parámetros	Una Lista Indexada inicializada
Efecto	Destruye a la estructura de datos subyacente de la Lista Indexada, liberando su espacio.
Requiere	Que la Lista Indexada alimentada como parámetro esté inicializada.
Modifica	La Lista Indexada ya no tiene a su estructura de datos subyacente inicializada, sino destruida

• Insertar

Nombre	Insertar
Parámetros	● Una Lista Indexada ● Un Elemento ● Una Posición (número entero como índice de inserción)
Efecto	Inserta en la estructura de datos subyacente al elemento tal que aquel elemento toma aquella posición
Requiere	Que: ● La Lista Indexada alimentada como parámetro esté inicializada. ● El Elemento alimentado como parámetro esté inicializado ● La Posición de inserción alimentada se refiera a un índice válido para la inserción: entre el principio (índice 0) y la posición hipotética después de la última (índice igual al tamaño de la lista)
Modifica	● La Lista Indexada crece en tamaño, en 1 unidad su cantidad de elementos. ● Si existe previamente un elemento en la posición de inserción, entonces aquel elemento preexistente y todos aquellos en posiciones siguientes (si existen) incrementan en 1 unidad su índice de posición. ● Si no existe previamente un elemento en la posición de inserción, entonces el elemento adopta esa posición y no modifica la posición de ningún otro.

• Borrar

Nombre	Borrar
Parámetros	● Una Lista Indexada ● Una Posición (número entero

	como índice de borrado)
Efecto	Elimina de la estructura de datos subyacente al elemento que posea aquella dirección.
Requiere	Que: ● La Lista Indexada alimentada como parámetro esté inicializada. ● La Lista Indexada alimentada como parámetro no esté vacía. ● La Posición de borrado alimentada se refiera a un índice válido para el borrado: entre el principio (índice 0) y el final (índice igual a una unidad menos que el tamaño de la lista)
Modifica	● La Lista Indexada disminuye en tamaño, en 1 unidad su cantidad de elementos. ● El elemento preexistente en la posición de borrado es destruido, y todos aquellos en posiciones siguientes (si existen) disminuyen en 1 unidad su índice de posición.

• Recuperar?

Nombre	Recuperar
Parámetros	● Una Lista Indexada ● Una Posición (número entero como índice de recuperado)
Efecto	Obtiene los datos de un Elemento que tenga la posición especificada como parámetro de la Lista Indexada.
Requiere	Que: ● La Lista Indexada alimentada como parámetro esté inicializada. ● La Lista Indexada alimentada como parámetro no esté vacía. ● La Posición de recuperado alimentada se refiera a un índice

	válido: entre el principio (índice 0) y el final (índice igual a una unidad menos que el tamaño de la lista)
Modifica	La Lista Indexada y su estructura de datos no sufren ningún cambio. Se retorna una copia del Elemento citado.

• Modificar

Nombre	Modificar
Parámetros	● Una Lista Indexada ● Un Elemento (para sustitución) ● Una Posición (número entero como índice para modificación)
Efecto	Reemplaza en la estructura de datos subyacente al elemento preexistente en aquella posición con el nuevo elemento alimentado.
Requiere	● La Lista Indexada alimentada como parámetro esté inicializada. ● El Elemento alimentado como parámetro esté inicializado ● La Posición de modificación alimentada se refiera a un índice válido para la inserción: entre el principio (índice 0) y el final (índice igual a una unidad menos que el tamaño de la lista)
Modifica	Únicamente el elemento referido por la posición (basado en un índice) es modificado: sus datos son reemplazados con los datos del elemento alimentado. El resto del modelo lógico y la estructura de datos subyacentes no sufren ningún cambio.

• NumElem?

Nombre	NumElem
Parámetros	● Una Lista Indexada

Efecto	Obtiene la cantidad de elementos existente en la Lista Indexada.
Requiere	• La Lista Indexada alimentada como parámetro esté inicializada.
Modifica	La Lista Indexada y su estructura de datos no sufren ningún cambio. Se retorna la cantidad de elementos existente en la Lista indexada.

Modelo Logico Arbol n-ario.

Árbol N-ario: El modelo lógico Árbol n-ario es un tipo de dato abstracto no lineal ya que está formado por una multitud de ramificaciones, de manera análoga a un árbol orgánico encontrado en la naturaleza.

Este modelo está conformado por Nodos y una jerarquía entre ellos. Se dice que cada Nodo, salvo la raíz (el Nodo más "arriba"), tiene solo un Nodo padre, pero cada Nodo puede tener n hijos (n-ario). Así, cada Nodo puede tener o una relación de preeminencia (padre-hijo), o de igualdad (hermanos).

• Crear

Nombre	Crear
Parámetros	Un Árbol
Efecto	Crea un Árbol y le asigna un espacio
Requiere	Un Árbol que no haya sido inicializado
Modifica	Se inicializa la estructura de datos subyacente.

• Destruir

Nombre	Destruir
Parámetros	Un Árbol
Efecto	Destruye la estructura de datos subyacentes del Árbol

Requiere	Un Árbol que haya sido inicializado
Modifica	La estructura del Árbol, libera el espacio usado

• PonerRaíz

Nombre	PonerRaíz
Parámetros	● Un Árbol ● Un Nodo
Efecto	Asigna al Nodo que se brinda como parámetro como la raíz del Árbol.
Requiere	● Un Árbol inicializado sin una raíz. ● Un Nodo no nulo
Modifica	Se asigna el Nodo brindado como la raíz del Árbol.

• AgregarHijo?

Nombre	AgregarHijo
Parámetros	● Un Árbol ● Una Etiqueta. ● Un Nodo “padre”
Efecto	Crea un Nodo que contiene la Etiqueta dada y lo agrega como hijo más izquierdo del Nodo “padre”
Requiere	● Un Árbol inicializado ● Un Nodo “padre” inicializado que hace parte del Árbol dado ● Una Etiqueta válida (que mantenga relación de tipos con los elementos del Árbol).
Modifica	Modifica la estructura del Árbol, agrega un hijo un Nodo del Árbol

• AgregarHijoMásDerecho?

Nombre	AgregarHijoMásDerecho

Parámetros	• Un Árbol • Una Etiqueta. • Un Nodo “padre”.
Efecto	Crea un Nodo que contiene la Etiqueta dada y lo agrega como hijo más derecho del Nodo “padre”
Requiere	Una Árbol inicializado, una Etiqueta válida, un Nodo “padre” inicializado que hace parte del Árbol dado
Modifica	Modifica la estructura del Árbol, pues agrega a un Nodo nuevo como hijo del Nodo alimentado.

• BorrarHoja

Nombre	BorrarHoja
Parámetros	Un Árbol, un Nodo
Efecto	Elimina una hoja de la estructura de datos
Requiere	Un Árbol inicializado, con al menos un Nodo, y que el Nodo que se brinda como parámetro sea una hoja.
Modifica	Disminuye en uno el tamaño del Árbol, elimina al Nodo que se brinda como parámetro.

• Raíz?

Nombre	Raiz
Parámetros	Un Árbol
Efecto	Devuelve el Nodo correspondiente a la raíz del Árbol. Si no hay raíz devuelve un Nodo nulo.
Requiere	Un Árbol inicializado.
Modifica	No modifica la estructura de datos.

• Padre?

Nombre	Padre
Parámetros	Un Árbol, un Nodo
Efecto	Devuelve un Nodo referente al padre de otro Nodo que nos brindan como parámetro, en caso de ser la raiz devuelve un Nodo nulo
Requiere	● Un Árbol inicializado, con al menos un Nodo. ● El Nodo que se pide como parámetro pertenece al Árbol
Modifica	No modifica la estructura de datos.

• HijoMásIzquierdo?

Nombre	HijoMasIzquierdo
Parámetros	Un Árbol, un Nodo
Efecto	Devuelve el hijo más izquierdo de un Nodo que se pide como parámetro. Si el Nodo dado no tiene hijo, entonces devuelve un Nodo nulo
Requiere	● Un Árbol inicializado con al menos un elemento. ● Que el Nodo que se pide de parámetro pertenezca al Árbol.
Modifica	No modifica la estructura de datos.

• Hermano Derecho?

Nombre	HermanoDerecho
Parámetros	Un Árbol, un Nodo
Efecto	Devuelve el hermano derecho del Nodo que se pide como parámetro. Si no existe un hermano derecho se devuelve un Nodo nulo
Requiere	● Un Árbol inicializado con al menos un Nodo

	• Que el Nodo que se pide de parámetro pertenezca al Árbol
Modifica	No modifica la estructura de datos.

• Etiqueta?

Nombre	Etiqueta
Parámetros	Un Árbol, un Nodo
Efecto	Devuelve la Etiqueta almacenada en un Nodo
Requiere	• Un Árbol inicializado. • Un Nodo inicializado que existe en el Árbol dado.
Modifica	No modifica la estructura de datos.

• ModificaEtiqueta

Nombre	ModificarEtiqueta
Parámetros	Un Árbol, un Nodo, una Etiqueta
Efecto	Cambia el valor de la Etiqueta de un Nodo por la Etiqueta dada
Requiere	• Un Árbol inicializado con al menos un Nodo. • Que el Nodo tenga una Etiqueta y exista en el Árbol
Modifica	Al Nodo que se brinda como parámetro

• NumNodos?

Nombre	NumNodos
Parámetros	Un Árbol, un Nodo válido
Efecto	Devuelve la cantidad de Nodos de un Árbol o subárbol. Se considera que la raíz de ese Árbol es el Nodo dado como parámetro. Si el Árbol solo tiene una raíz y ningún otro Nodo entonces

	devuelve 1.
Requiere	Un Árbol (o sub árbol) validado e inicializado
Modifica	No modifica la estructura de datos.

5. Manual del Usuario

5.1. Requerimientos de Hardware

Este programa fue desarrollado en una computadora con las siguientes especificaciones: CPU Intel(R) Core(TM) i3-1005G1 CPU @ 1.20GHz 1.19 GHz, con 8GB de memoria RAM.

5.2. Requerimientos de Software

Para la ejecución de este programa se requiere tener instalada una versión de c++ de la 11.1.0 en adelante y un medio o ambiente para la ejecución del programa en c++.

5.3. Arquitectura del programa

El programa consta de un archivo “main.cpp”, el cual está encargado de mostrar la interfaz al usuario.

Para la implementación de la estructura cola se utilizan los siguientes archivos:

- “Cola.cpp”: Está encargado de implementar la cola mediante arreglo circular.

Para la implementación de la estructura lista Indexada se utilizan los siguientes archivos:

- “Celda.h”: Está encargado de implementar los métodos de la clase celda, estos serán necesarios para la creación de una lista simplemente enlazada.
- “Lista.h”: Está encargado de implementar los métodos de la clase lista, que en conjunto con “Celda.h” crean una lista simplemente enlazada.
- “ListaIndexada.h”: Header file de la lista indexada, implementa los métodos de la clase.
- “ListaIndexada.cpp”: Está encargado de implementar los operadores básicos de la clase lista indexada, que en conjunto con “Celda.h” crean una lista simplemente enlazada.

Para la implementación del árbol con arreglo con señalador al padre se usan los siguientes archivos:

- “3.1.hpp”: Clase utilizada para la estandarización de nodos entre los diferentes árboles.
- “Arbol_1.cpp”: Clase utilizada para implementar los operadores basicos del arbol.

Para la implementación del arbol lista de hijos:

- “3.2.hpp”: Clase utilizada para la estandarización de nodos entre los diferentes árboles.
- “ArbolLH.hpp”: Está encargado de implementar los operadores básicos de la clase árbol, que en conjunto con “Celda.h” y “Lista.h”, crean una estructura lista de hijos.
- “Celda.h”: Está encargado de implementar los métodos de la clase celda, estos serán necesarios para la creación de una lista simplemente enlazada.
- “Lista.h”: Está encargado de implementar los métodos de la clase lista, que en conjunto con “Celda.h” crean una lista simplemente enlazada.
- “NodoConcreto.hpp”: Clase utilizada como Nodo del arbol n-ario.

Para la implementación del árbol con Hijo Mas Izquierdo-Hermano Derecho con contador:

- “3.3.hpp”: Clase utilizada para la estandarización de nodos entre los diferentes árboles.
- “Arbol.hpp”: Clase encargada de implementar los operadores básicos del modelo lógico árbol, como característica importante se menciona que esta clase posee contador.

Para la implementación del árbol con Hijo Mas Izquierdo-Hermano Derecho con puntero al padre y al hermano izquierdo sin contador:

- “3.4.hpp”: Clase utilizada para la estandarización de nodos entre los diferentes árboles.
- “Arbol.hpp”: Clase encargada de implementar los operadores básicos del modelo lógico árbol, como característica importante se menciona que esta clase no posee contador y posee un señalador al padre y al hermano izquierdo.

Para la implementación del árbol con Hijo Mas Izquierdo-Hermano Derecho tal que el último hijo de un nodo apunta al padre sin contador:

- “3.5.hpp”: Clase utilizada para la estandarización de nodos entre los diferentes árboles.
- “Arbol.hpp”: Clase encargada de implementar los operadores básicos del modelo lógico árbol, como característica importante se menciona que esta clase no posee contador y el ultimo hijo de un nodo apunta al padre.

5.4. Compilación

Para todos nuestros códigos usamos el mismo compilador: G++.

Cola

Note: en nuestro caso la función main esta metida dentro del mismo archivo Cola

Para compilar la Cola, usamos el siguiente comando:

g++ Cola.cpp -o code.out

Lista indexada

Todos los archivos se encuentran en el mismo repositorio:

*g++ *.cpp*

Árboles

Para cada prueba se usa el mismo main, solo se descomenta uno de los header que contiene el árbol que queremos usar, y después se compila con el siguiente comando:

g++ main.cpp

5.5. Especificación de las funciones del programa

Los métodos recorrer y buscar de los árboles se realizan en preorden.

6. Datos de Prueba

6.1. Formato de los datos de prueba

6.2. Salida esperada

6.3. Salida obtenida (Análisis en caso de fallo)

8. Listado de Archivos (Estructura de las Carpetas)

9. Referencias o Bibliografía

Arquitectura del programa

Se especifica la forma en la que está diseñada la aplicación, las capas que lo componen y la forma de comunicación, por ejemplo si se tiene capas como interfaz, lógica del programa, estructuras de datos, almacenamiento y acceso de archivos, entre otros.

Compilación

Se especifica el compilador utilizado y la forma de compilar y ejecutar el programa. Se especifican restricciones y cambios que hay que hacer en el código para poder compilar, utilizar y probar la aplicación.

Especificación de las funciones del programa

Se especifican las funciones de cada una de las opciones que presenta la interfaz, se

enumera cada una de ellas, la forma en la que se le ingresan los datos, sus salidas, y sus restricciones.

Listado de Archivos (Estructura del Disco)

Ejemplo:

El archivo etapaX_eaXXXXX_eaXXXXX.zip contiene:

Una carpeta llamada etapaX_eaXXXXX_eaXXXXX.zip la cual contiene:

- Una Carpeta Bin: Contiene la aplicación en el archivo ejecutable

-----Controlador.exe, Controlador.out

-Una Carpeta src: Contiene el código fuente C++.

-----Estructura X

--Archivos .cpp, archivos .h

-----Estructura X

-- Archivos .cpp, archivos .h

-Una Carpeta doc : Contiene la documentación externa y el manual de usuario del programa.

-----Documentacion.doc