Descripción de cada TDA, indicando las pre y postcondiciones de cada una de las operaciones

**TDA Árbol:**

Destructor y constructor del TDA arbol

Arbol ();

~Arbol ();

//pre: debe ser un dato valido

//post: Inserta un dato en árbol ordenado

void insertar(Dato dato);

//pre: deben existir ambos nodos

//post: swapear nodos

void swap(Nodo\* nodo\_1,Nodo\* nodo\_2);

//pre: debe ser un dato valido

//post: borra dato del arbol sin devolver el dato

//void borrar\_sin\_devolver (Dato dato);

//Buscar nodo

//pre: debe ser un dato valido

//post: devuelve el nodo del dato a buscar

Dato buscar(Dato dato);

// Comprobar si es un nodo hoja:

//pre: debe existir el nodo a comprobar

//post: Comprueba si es un nodo hoja:

bool es\_hoja(Nodo\* r)

//pre: debe existir el nodo

//post: obtiene el nodo izquierdo o derecho ,elimina el nodo y lo setea en null

void Podar(Nodo\* nodo);

//pre: debe existir el arbol

//post: llama a mostrar nodos para mostrar ordenadamente el arbol

void mostrar\_arbol();

TDA Nodo:

PRE: Ninguna

POST: Crea un nodo con el dato d y el puntero a NULL

Nodo(Dato d);

Destructor

PRE: Nodo creado

POST: No hace nada

~Nodo();

Setea el dato (lo cambia)

PRE: el nodo tiene que estar creado , d tiene que ser un dato válido

POST: el nodo queda con el dato d

void agregar\_dato(Dato d);

Setear el puntero al nodo izquierdo

PRE: nodo creado y puntero válido

POST: el puntero al siguiente apuntará a izq

void agregar\_izquierdo(Nodo\* izquierdo);

Setear el puntero al nodo derecho

PRE: nodo creado y puntero válido

POST: el puntero al siguiente apuntará a der

void agregar\_derecho(Nodo\* derecho);

Obtener el dato

PRE: nodo creado

POST: devuelve el dato que contiene el nodo

Dato obtener\_dato();

Obtener el puntero al nodo izquierdo

PRE: nodo creado

POST: Devuelve el puntero al siguiente nodo,asi es el último devuelve NULL

Nodo\* obtener\_izquierdo();

Obtener el puntero al nodo derecho

PRE: nodo creado

POST: Devuelve el puntero al siguiente nodo , si es el último devuelve NULL

Nodo\* obtener\_derecho();

¿Hay un siguiente?

PRE: nodo creado

POST: V si tiene sig. F sino

bool tiene\_siguiente();

**TDA Cliente:**

//Constructor

Cliente ();

//Constructor con parametros de tipo Individuo

Cliente (string nombre, int teléfono, int precio);

//Constructor con parametros de tipo Familia

Cliente (Lista &nom, int teléfono, int precio);

//pre: -

//post: imprime por pantalla los clientes

void mostrar\_cliente ();

//pre: debe ser un número valido

//post: agregar un teléfono al cliente

void modificar\_telefono (int tel);

// PRE: -

// POST: obtiene el telefono

virtual int devolver\_telefono ();

virtual void mostrar\_datos (){}

//Destructor

virtual ~Cliente ();

**TDA Individuo:**

//Constructor sin parametros

Individuo();

//Constructor con parametros de Individuo

Individuo(string nom);

//Destructor

~Individuo();

**TDA Familia:**

//Constructor sin parametros

Familia();

//Constructor con parametros

Familia(Lista &nom);

//Destructor

~Familia();

**TDA Lista:**

Constructor

PRE: Ninguna

POST: construye una lista vacía

- primero apunta a nulo

- tam = 0

Lista();

Destructor

PRE: Lista creada

POST: Libera todos los recursos de la lista

~Lista();

Agregar un elemento a la lista

PRE: lista creada y d válido

POST: agrega un dato dentro de un nodo al principio

- modifica el primero

- tam se incrementa en 1

void insertar(Dato d);

Obtener el tamaño de la lista

PRE: Lista creada

POST: devuelve el tamaño de la lista (cantidad de nodos)

unsigned devolver\_tamanio();

Obtener el dato que está en la posición pos

PRE: - lista creada y no vacía

- 0 < pos <= tam

POST: devuelve el dato que está en la posición pos

se toma 1 como el primero

Dato devolver\_dato(unsigned pos);

¿Lista vacia?

PRE: Lista creada

POST: T si es vacia, F sino

bool lista\_vacia();

Borrado del nodo que está en la posición pos

PRE: - lista creada y no vacía

- 0 < pos <= tam

POST: libera el nodo que está en la posición pos

se toma 1 como el primero

void eliminar\_dato(unsigned pos);

TDA Nodo Lista:

Constructor con parametro

PRE: Ninguna

POST: Crea un nodo con el dato d y el puntero a NULL

Nodo(Dato d);

Destructor

PRE: Nodo creado

POST: No hace nada

~Nodo();

Setea el dato (lo cambia)

PRE: el nodo tiene que estar creado

d tiene que ser un dato válido

POST: el nodo queda con el dato d

void agregar\_dato(Dato d);

Setear el puntero al siguiente nodo

PRE: nodo creado y ps válido

POST: el puntero al siguiente apuntará a ps

void agregar\_siguiente(Nodo\* ps);

Obtener el dato

PRE: nodo creado

POST: devuelve el dato que contiene el nodo

Dato devolver\_dato();

Obtener el puntero al nodo siguiente

PRE: nodo creado

POST: Devuelve el puntero al siguiente nodo

si es el último devuelve NULL

Nodo\* devolver\_siguiente();

¿Hay un siguiente?

PRE: nodo creado

POST: V si tiene sig. F sino

bool tiene\_sig(); };

**Tda Menu:**

//Constructor

Menu();

//pre: debe existir el arbol clientes y debe ingresar un precio valido

//post: agregar al arbol un nuevo cliente con su precio base

void agregar\_cliente(Arbol &abb,int precio);

//pre: debe existir el arbol de clientes

//post: imprime por pantalla los datos del cliente que desea buscar el usuario

void buscar\_cliente(Arbol &abb);

//pre: -

//post: Imprime el menu del sistema

void imprimir();

//pre: debe existir el arbol clientes

//post: elimina el cliente a travez del numero ingresado por el usuario

//void eliminar\_cliente(Arbol &abb);

//pre: Las listas deben existir

//post: Ejecuta el menu a travez de una lista

void ejecutar(Arbol &abb,int precio);

//Destructor

~Menu();

**TDA Archivo:**

// PRE:recice un arbol abb

// POST: agrega a una lista los familiares/individuos y muestra por pantalla si ocurrio un error al abrir el archivo

void cargar\_archivo(Arbol &abb,int precio);

// PRE: recibe un arbol abb, y un archivo clientes abierto

// POST: agrega a una lista los familiares/individuos

void cargar\_archivo\_clientes(Arbol &abb,ifstream &clientes\_txt,int precio);