TAREA #2 **TIPOS DE DATOS ABSTRACTOS**

CINTHIA GUADALUPE OLIVAS CALDERON NO.17212165

Un TDA permite representar un concepto a través de la definición de sus características (datos que lo conforman) y de sus operaciones (algoritmos que manipulan los datos que lo conforman). La implementación de un TDA en un lenguaje de programación usualmente se sirve de un tipo de dato estructurado para la definición de los datos que lo conforman

Se denomina abstracto ya que la intención es que quien lo utiliza, no necesita conocer los detalles de la representación interna o bien el cómo están implementadas las operaciones.

Tipos de Datos Abstractos

Como ya se revisó, un TDA se refiere al concepto matemático básico que define a un tipo de dato. Están formados por los datos (estructuras de datos) y las operaciones (procedimientos o funciones) que se realizan sobre esos datos.

TDA = Valores +operaciones

El diseño de un TDA puede ser especificado utilizando ya sea un enfoque informal, como un enfoque formal.

Especificación informal:

Describe en lenguaje natural todos los datos y sus operaciones, sin aplicar conceptos matemáticos complicados para las persona que no están familiarizados con los TADs, de manera, que todas las personas que no conocen a fondo las estructura de los TADs, lo puedan entender de manera sencilla y que esas mismas personas puedan explicarlo de la misma manera natural, a todos con la misma facilidad con la que ellos lo entendieron.

Especificación Formal: Una de las ventajas de especificar formalmente el diseño de un TDA es que permite la posibilidad de simular especificaciones a través de la definición de precondiciones y postcondiciones para las operaciones de los TDAs.

- **El tipo de datos abstracto «*lista*»**

Colección de elementos homogéneos (del mismo tipo: TipoElemento) con una relación LINEAL establecida entre ellos. Pueden estar ordenadas o no con respecto a algún valor de los elementos y se puede acceder a cualquier elemento de la lista.

Operaciones:

TipoLista Crear()

void Imprimir(TipoLista lista)

int ListaVacía(TipoLista lista)

void Insertar(TipoLista lista, TipoElemento elem)

void Eliminar(TipoLista lista, TipoElemento elem)

Opcionalmente, si la implementación lo requiere, puede definirse:

int ListaLlena(TipoLista lista)

Hay que tener en cuenta que la posición de la inserción no se especifica, por lo que dependerá de la implementación. No obstante, cuando la posición de inserción es la misma que la de eliminación, tenemos una subclase de lista denominada pila, y cuando insertamos siempre por un extremo de la lista y eliminamos por el otro tenemos otra subclase de lista denominada cola.

Especificación del TDA Cola

Colección de elementos homogéneos (del mismo tipo: TipoElemento) ordenados cronológicamente (por orden de inserción) y en el que sólo se pueden añadir elementos por un extremo (final) y sacarlos sólo por el otro (frente). Es una estructura FIFO (First In First Out).

Operaciones

```
 TipoCola Crear();

/* Crea y devuelve una cola vacía */

 int ColaVacía(Tipocola Cola);

/* Devuelve 1 sólo si “cola” está vacía */

 int ColaLlena(TipoCola Cola);

/* Devuelve 1 sólo si “cola” está llena */

int Sacar(Tipocola Cola, TipoElemento *elem);

 /* Saca el siguiente elemento del frente y lo pone en “elem” */

 int Meter(TipoCola Cola, TipoElemento elem);

 /* Mete “elem” al final de la cola */
```

Especificación del TAD Pila TipoPila

Colección de elementos homogéneos (del mismo tipo: TipoElemento) ordenados cronológicamente (por orden de inserción) y en el que sólo se pueden añadir y extraer elementos por el mismo extremo, la cabeza. Es una estructura LIFO (Last In First Out)

Operaciones

```
TipoPila Crear(void); /* vacía */

 /* Crea una pila vacía */

 int PilaVacia(TipoPila pila);

/* Comprueba si la pila está vacía */

int PilaLlena(TipoPila pila);
```

/* Comprueba si la pila está llena */

void Sacar(TipoPila *pila, TipoElemento *elem);

/* Saca un elemento. No comprueba antes si la pila está vacía */

void Meter(TipoPila pila, TipoElemento elem);

/* Mete un elemento en la pila. No comprueba si está llena. */

Especificación del TAD Árbol binario ·

Estructura de elementos homogéneos (del mismo tipo: TipoElemento) con una relación JERARQUICA establecida entre ellos a modo de árbol binario.

Operaciones

CrearVacío(): TipoABin

(* Crea un árbol vacío *)

CrearRaíz(elem: TipoElemento): TipoABin

(* Crea un árbol de un único elemento *)

ArbolVacío(árbol: TipoABin): B (* Comprueba si “árbol” está vacío *)

AsignarIzq(VAR árbol: TipoABin; izq: TipoABin)

(* Establece “izq” como subárbol izq. de “árbol” *)

AsignarDer(VAR árbol: TipoABin; der: TipoABin)

(* Establece “izq” como subárbol izq. de “árbol” *)

Info(árbol: TipoABin): TipoElemento

(* Devuelve el nodo raíz de “árbol” *)

Izq(árbol: TipoABin): TipoABin

(* Devuelve el subárbol izquierdo de “árbol” *)

Der(árbol: TipoABin): TipoABin

(* Devuelve el subárbol derecho de “árbol” *)

BorrarRaíz(VAR árbol, izq, der: TipoABin)

(* Devuelve en “izq” y “der” los subárboles de “árbol” *)

Imprimir(árbol: TipoABin)

(* Imprime en pantalla el contenido completo de “árbol” *)

Obsérvese que no se define una operación de inserción de elementos ya que el punto de inserción puede variar de una aplicación a otra, razón por la cual se deja al usuario del TAD que defina su propio Insertar ().

En Imprimir () habría que definir de alguna manera el orden de impresión.

Tipo ABB

Estructura de elementos homogéneos (del mismo tipo: TipoElemento) con una relación JERARQUICA establecida entre ellos a modo de árbol binario en el que el subárbol izquierdo de cualquier nodo, si no está vacío, contiene valores menores, con respecto a algún campo clave de tipo TipoClave, que el valor que contiene dicho nodo, y el subárbol derecho, si no está vacío, contiene valores mayores.:

REFERENCIAS BIBLIOGRÁFICAS

José Fager W. Libardo Pantoja Yépez Marisol Villacrés Luz Andrea Páez Martínez Daniel Ochoa Ernesto Cuadros-Vargas. (2014). Introducción a las estructuras de datos. En Estructuras de datos(222). Latinoamérica: LATIn.

Leiva Olivencia, Jose Luis. (2003). Apuntes de Diseño de Algoritmos. 2004, de Departamento de Lenguajes y Ciencias de la Computación Sitio web: http://www.lcc.uma.es/~jlleivao/algoritmos/t5.pdf