Abraham Magaña Hernández - 220791217

DIA 29 MES 01 AÑO 2022

• Investigación 03: Operaciones TDA Lista Simplemente Ligada

- void Insertar(data): Esta función sirve para insertar un elemento (data) dentro de un nodo y ingresarlo a la lista.

- void Insertar_final(data): Esta operación condiciona un nodo aux en su puntero siguiente hasta que sea diferente de nullptr y ingresa ahí el nodo con el elemento (data).

- void Insertar_pos(data, pos): Esta operación recorre todos los nodos hasta llegar a la posición deseada, una vez hecho eso ingresa un nuevo nodo con el elemento (data)

- void eliminar(data): Eliminar es algo complejo, para empezar necesita dos nodos aux, uno para recorrer la lista, otro llamado anterior, sí usando una condicional data == aux->data, hacemos la cabecera header = header->siguiente, sino recorremos la lista mientras data != aux->siguiente->data, una vez hecho esto saltamos el nodo a eliminar anterior = aux, aux = aux->siguiente y anterior->siguiente = aux->siguiente. Finalmente delete aux.

- void eliminar_todo( ): Eliminar todo, crea un nodo aux para recorrer ela lista, si aux contiene algo aux = header, header = nullptr, delete aux.

- Nodo buscar(data): Creamos un nodo aux para recorrer ela lista, mientras data != aux->data, ahora si data == aux->data, return &aux;

- void Inicializa(): Usando this->header = nullptr, inicializamos el nodo que contiene nuestra lista.

- bool Vacia(): Usando el condicional comprobamos si header != nullptr, entonces return false;, sino return true.

- data Primero(): Usando un condicional si header != nullptr, return header->data.

- data Ultimo(): Recorremos la lista mientras aux->siguiente, aux = aux->siguiente, finalmente return aux->data.

Abraham Magaña Hernández - 220791217

DIA 29 MES 01 AÑO 2022

- int Tamaño() { Creamos un contador en 0, luego se Vacía() regresamos 0, sino mientras aux, contador ++, aux -> siguiente. return contador.

- data siguiente(data) { Creamos un nodo aux para recorrer la lista, mientras data != aux -> data, luego si data == aux -> data y también si aux -> siguiente != nullptr, return aux -> siguiente -> data.

- data Anterior(data) { Creamos un nodo aux para recorrer la lista mientras data != aux -> siguiente -> data, una vez llegado solo return aux -> data.

- void mostrarTodo() { Creamos un nodo aux para recorrer ela lista mientras aux != nullptr vamos imprimiendo por pantalla { std:: cout << "Data :" << aux -> data << std:: endl, aux = aux -> siguiente.

• Referencias:

- Abraham Magaña Hernández. (2022). Lista.h, 28-01-2022, de GitHub, Web: github.com/Thains_H/s-data-structures-1/blob/main/code/TDA%20-%20Lista%20simple/Lista.h