2-11-2021

Lista dinamica

Practica: 5

Materia: Estructura de datos

Sección: D01.

Código: 216584703

Carrera: Ingeniería en computación.

Nombre alumno: Padilla Pérez Jorge Daray

Nombre profesor: Julio Esteban Valdes Lopez





Introducción

Mi práctica consiste en la implementación de una lista dinámica en la cual se implementan lo típico de un TDA Lista lo que viene siendo insertar en orden como si fuera una lista consultar elemento remover etc. En esta practica se logro hacer todo lo pedido para esta, además de poder realizar la función de localizar posición por elemento la cual me costó más, pero si se logró.

También aunque no se implementaron las funciones anterior y siguiente creo que es una función fácil de comprender y de hacer por lo que me parece bien que se hayan saltado esas 2 funciones que en si no existe una utilidad buena para ellos.

Pantallazos

Texto

Descripción generada automáticamente

Aquí se muestra el menú

Texto

Descripción generada automáticamente

Aquí se aprecia que se introducen bien los datos y funciona bien la función mostrar

Texto

Descripción generada automáticamente

Aquí se localiza por posición el dato

Texto

Descripción generada automáticamente

Aquí se recupera la posición de forma lineal

Texto

Descripción generada automáticamente

Aquí se elimina la posición 1 de la lista

Conclusión

Respecto a la realización del código concluyo que se logró bien la realización de este programa ya que a lo que se puede apreciar funciona de manera correcta como debería hacerlo una lista además de implementar bien sus funciones recalco mis conocimientos, y estoy abierto a posibles errores que pueda tener el programa al momento de que el profe la evalúe.

También se concluye que aunque no parezca que tiene una utilidad buena al momento de referirse a trabajo si lo piensas un rato le hayas muchas utilidades importantes tanto que pensándolo muchas aplicaciones y juegos realizan una lista implementada en estos mismos por lo cual espero poder aprender mas sobre las listas y los tipos de datos abstractos

**Código fuente**

**#include<stdio.h>**

**#include<stdlib.h>**

**#include <string.h>**

**struct node{**

**int data;**

**struct node \*next;**

**int dato;**

**};**

**struct Lista{**

**void inicializa();**

**int inserta();**

**void Localiza();**

**void Imprime();**

**void Recupera();**

**void eliminateNode();**

**void anular();**

**node \*first;**

**node \*last;**

**Lista(){**

**inicializa();**

**}**

**};**

**void Lista::inicializa(){**

**first = nullptr;**

**last = nullptr;**

**};**

**Lista mi\_lista;**

**int main()**

**{**

**int opc;**

**do{**

**printf("\tINTRODUCE AN OPCION:\n");**

**printf("\n0.-Salir\n1.-Introduce\n2.-Localiza\n3.-Muestra\n4.-Recupera\n5.-Elimina\n6.-Anular\n");**

**mi\_lista.Imprime();**

**scanf("%d", &opc);**

**switch(opc){**

**case 0: printf("\tAdios \n");**

**break;**

**case 1: printf("\tInsert a node in the list\n\n");**

**mi\_lista.inserta();**

**break;**

**case 2: printf("\tSeek a node in the list\n\n");**

**mi\_lista.Localiza();**

**system("pause");**

**break;**

**case 3: printf("\tShow the list\n\n");**

**mi\_lista.Imprime();**

**system("pause");**

**break;**

**case 4: printf("\tRecuperate the list\n\n");**

**mi\_lista.Recupera();**

**system("pause");**

**break;**

**case 5: printf("\tEliminate node\n\n");**

**mi\_lista.eliminateNode();**

**system("pause");**

**break;**

**case 6: printf("\tAnulate List\n\n");**

**mi\_lista.anular();**

**system("pause");**

**break;**

**default: printf("\tChoose a correct value");**

**break;**

**}**

**system("cls");**

**}while(opc!=0);**

**return 0;**

**}**

**int Lista::inserta()**

**{**

**node \*new\_n = (node \*) malloc(sizeof(node));**

**if(!new\_n){// new\_==NULL**

**printf("Memory allocation error, new node could not be created");**

**return -1;**

**}**

**printf("INTRODUCE THE VALUE OF THE NEW NODE: ");**

**scanf("%d", &new\_n -> data);**

**//se guardara en el nuevo nodo en su propiedad dato**

**if(first == NULL){**

**first = new\_n;**

**first ->next = NULL;**

**last = new\_n;**

**}**

**else{**

**last -> next = new\_n;**

**new\_n -> next = NULL;**

**last = new\_n;**

**}**

**printf("Dato: ");**

**scanf("%d",&new\_n->dato);**

**printf("\nTHE NODE HAVE BEEN INTRODUCE CORRECTLY\n\n");**

**system("pause");**

**return 0;**

**}**

**void Lista::Imprime(){**

**node \*temp = (node \*) malloc(sizeof(node));**

**temp = first;**

**if (first != NULL){**

**while (temp != NULL){**

**printf("Posicion %d ->", temp->data);**

**printf("Dato: %i\n", temp->dato);**

**temp = temp ->next;**

**}**

**}**

**else{**

**printf("\n The list is empty\n");**

**}**

**}**

**void Lista::eliminateNode()**

**{**

**node \*actual = (node \*) malloc(sizeof(node));**

**actual = first;**

**node\* before = (node \*) malloc(sizeof(node));**

**before = NULL;**

**int soughtnode = 0, found = 0 ;**

**printf(" Introduce the posicion of the node that you want to eliminate: ");**

**scanf("%d", &soughtnode);**

**if(first != NULL){**

**while(actual != NULL && found != 1){**

**if(actual -> data == soughtnode){**

**if(actual == first){**

**first = first ->next;**

**}**

**else{**

**before -> next = actual -> next;**

**}**

**printf("\nThe link of the node have been eliminated");**

**found = 1;**

**}**

**before = actual;**

**actual = actual ->next;**

**}**

**if(found == 0){**

**printf("\nthe node was not found\n\n");**

**}**

**else{**

**//->->->->->->->->->->->->->->->->->->**

**free(before);**

**printf("\n\nNODE ELIMINATED SUCCESSFUL");**

**}**

**}**

**else{**

**printf("\nTHE LIST IS EMPTY\n\n");**

**}**

**printf("\n");**

**}**

**void Lista::Localiza(){**

**node \*temp = (node \*) malloc(sizeof(node));**

**temp = first;**

**int soughtnode = 0, found = 0 ;**

**printf(" Introduce the posicion of the node that you want to seek: ");**

**scanf("%d", &soughtnode);**

**if (first != NULL){**

**while (temp != NULL && found != 1){**

**if ( temp -> data == soughtnode ){**

**printf("Posicion %d", temp->data);**

**printf("Dato: %i\n\n", temp->dato);**

**found = 1;**

**}**

**temp = temp ->next;**

**}**

**if (found == 0){**

**printf("El nodo no fue encontrado");**

**}**

**}**

**else{**

**printf("\n The list is empty\n");**

**}**

**}**

**void Lista::Recupera(){**

**node \*temp = (node \*) malloc(sizeof(node));**

**temp = first;**

**int soughtnode = 0, found = 0 ;**

**printf(" Introduce the dato of the node that you want to recuperate: ");**

**scanf("%d", &soughtnode);**

**if (first != NULL){**

**while (temp != NULL && found != 1){**

**if (temp -> dato == soughtnode){**

**printf("Posicion %d", temp->data);**

**printf("Dato: %i\n\n", temp->dato);**

**found = 1;**

**}**

**temp = temp ->next;**

**}**

**if (found == 0){**

**printf("El nodo no fue encontrado");**

**}**

**}**

**else{**

**printf("\n The list is empty\n");**

**}**

**}**

**void Lista::anular()**

**{**

**node \*actual = (node \*) malloc(sizeof(node));**

**actual = first;**

**node\* before = (node \*) malloc(sizeof(node));**

**before = NULL;**

**if(first != NULL){**

**while(actual != NULL){**

**if(actual == first){**

**first = first ->next;**

**}**

**else{**

**before -> next = actual -> next;**

**}**

**before = actual;**

**actual = actual ->next;**

**}**

**free(before);**

**printf("\n\nNODE ELIMINATED SUCCESSFUL");**

**}**

**else{**

**printf("\nTHE LIST IS EMPTY\n\n");**

**}**

**printf("\n");**

**}**