30-9-2021

Lista estática mejorada

Practica: 3

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 estructurada 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 se pudo lograr recorrer los datos una vez ingresados en la lista, 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.

Además de la practica 2 se agregaron métodos de búsqueda lineal y binario, también de métodos de ordenamiento tanto iterativos en este caso (3), y recursivos en este caso (2).

Pantallazos

Texto

Descripción generada automáticamente

Estructura de la lista junto con el tamaño de la lista y donde se implementan todas las funciones del programa.

Texto

Descripción generada automáticamenteFunciones principales de una lista, mas el constructor de la lista.

Pantalla de computadora con letras

Descripción generada automáticamente con confianza media

Funcion insertar en la cual se ponen por posicion, cuidando la continuidad de esta, ademas de recorrer los datos de la lista si se inserta en una posicion anterior al ultimo.

Pantalla de computadora con letras

Descripción generada automáticamente con confianza media

Funcion elimina la cual simplemente hace que los saque de la lista y ultimo disminuye al eliminar el elmento de la posicion dada

Texto

Descripción generada automáticamente

Función imprimir no tiene mayor complejidad mas que un for que vaya imprimiendo los datos conforme las posiciones.

Texto

Descripción generada automáticamente

Función recupera recupera el dato guardado en la posición que se le solicita, simplemente recibe la posición que se agrega en el menú y esta se imprime.

Texto

Descripción generada automáticamente

Función booleana que revisa si la lista esta ordenada empezando por el primer dato de la lista y va comparando con el siguiente si es mayor.

Texto

Descripción generada automáticamente

Función localiza que recupera la posición conforme al dato, aquí nomas se aprecia el menú para el usuario.

Texto

Descripción generada automáticamente

Case 1 de la función recupera la cual es el método lineal y es tan fácil como buscar el dato a través de toda la lista y si se encuentra el dato que puso el usuario se imprime si no se pone dato no encontrado.

Texto

Descripción generada automáticamente

Case 2 en el cual se manda a llamar a la función de lista ordenada para saber si la lista esta ordenada ya que este método binario así lo requiere, una vez se comprobó que si se inicializa la función.

Texto

Descripción generada automáticamente

Funcion metodo de ordenamiento burbuja(iterativo), el cual hace parecido como la manera de validar si la lista esta ordenada pero en vez de dar aviso intercambia los valores para que queden acomodados.

Texto

Descripción generada automáticamente

Funcion metodo de ordenamiento insercion(iterativo), el cual inicia el iterador en el primer dato y se garega un auxiliar para poder darle condicion al mientras, el cual se encarga de cambiar los datos por su posicion correcta.

Texto

Descripción generada automáticamente

Función método de ordenamiento selección(iterativo).

Texto

Descripción generada automáticamente

Función método de ordenamiento mezcla(recursivo), el cual recibe de parámetro el extremo izquierdo y el derecho de la lista en este caso es igual a primero y ultimo de esta, para este método se copia la lista, por lo que esta en la estructura principal de la lista.

Texto

Descripción generada automáticamente

Función recursiva Quicksort(recursiva), la cual al igual que mezcla recibe el extremo derecho e izquierdo.

Texto

Descripción generada automáticamente

Se inicializa una variable mi\_lista de la estructura principal Lista, ademas de un do-while el cual muestra el menu principal y hace repetir el ciclo hasta que se ponga un 12 para la salida.

Texto

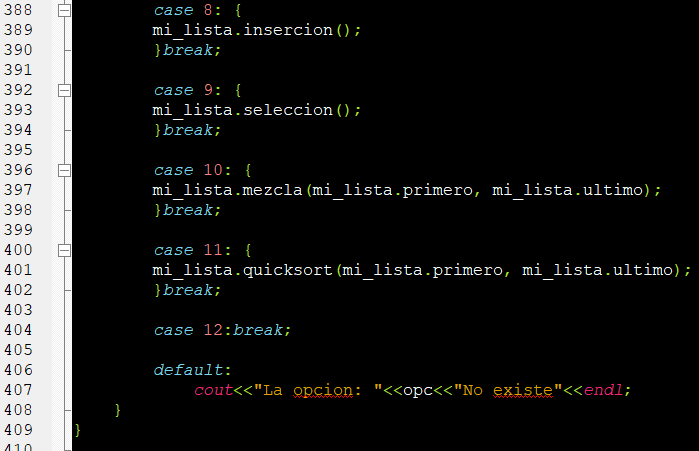
Descripción generada automáticamente

El menú no tiene mayor complicación.

Texto

Descripción generada automáticamente

Las opciones del menu.



Final del código explicado.

Texto

Descripción generada automáticamente

Menu de opciones para el usuario.

Texto

Descripción generada automáticamente

Se insertan elementos en la lista y se imprimen.

Texto

Descripción generada automáticamenteSe elimina la posicion 2 de la lista en este caso 75

Texto

Descripción generada automáticamente

Al mostrarlos nuevamente se aprecia que ya fue eliminado.  
  
Texto

Descripción generada automáticamenteAl recuperar el elemento de la posicion 3 arroja el dato junto la psocion buscada siguiendo la lista que tenemos en este caso el dato 44.

Texto

Descripción generada automáticamenteEn este caso al escoger la opcion 4 pide el dato que queremos localizar.

Texto

Descripción generada automáticamenteUna vez hecho eso nos da la opcion de escoger de que metodo encontrar el dato, en este caso se escoge la primera opcion, la cual arroja la posicion en la que se ecuentra y refairma el dato que contiene abajo.

Texto

Descripción generada automáticamenteSi ahorita mismo queremos buscarlo por metodo binario no se puede ya que necesita estar ordenada la lista para esto y manda el mensaje.

Texto

Descripción generada automáticamenteHacemos un ordenamiento de los datos que tenemos con cualquier metodo de los 5, los 5 funcionan.

Texto

Descripción generada automáticamenteY ahora si encuntra el valor ya que si esta ordenada la lista.

Interfaz de usuario gráfica, Texto

Descripción generada automáticamenteSi ahora insertamos un elemnto en la posicion 0 se recorren los valores despues de este, y ademas volvemos a desordenar la lista.

Texto

Descripción generada automáticamenteaquí se muestra lo que se explico.

Texto

Descripción generada automáticamenteEntonces si queremos buscar el mismo dato “44”, por metodo binario ya no esta ordenada nuevamente y manda el mensaje.

Texto

Descripción generada automáticamenteVolvemos a ordenar la lista con cualquier metodo.

Texto

Descripción generada automáticamentey ahora si la encuentra pero ahora en la posicion 2, no como al inicio en la posicion 1, ya que el dato 20 ahora acupa ese lugar.

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.

Al momento de realizar el uso de métodos de ordenamientos y de búsquedas mejoro muchísimo el código base de la practica 2 y se le agregan más utilidades además de lograr un nuevo reto que fue su implementación.

**Código fuente**

**#include <iostream>**

**#include <cstring>**

**#define TAMLISTA 10**

**using namespace std;**

**void menu();**

**typedef int tipo\_dato,temp;**

**struct Lista{**

**tipo\_dato datos[TAMLISTA];**

**void inicializa();**

**bool vacia();**

**bool llena();**

**void insertar(int pos, tipo\_dato elem);**

**void elimina(int pos);**

**int ultimo;**

**int primero;**

**void localiza(tipo\_dato dato);**

**void recupera(int pos);**

**void imprimir();**

**void anular();**

**void burbuja\_mejorada();**

**void insercion();**

**void seleccion();**

**void mezcla( int , int );**

**void quicksort ( int , int );**

**bool lista\_ordenada ();**

**temp arreglo\_copia[TAMLISTA];**

**Lista(){**

**inicializa();**

**}**

**};**

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

**ultimo = -1;**

**primero = 0;**

**}**

**bool Lista::vacia(){**

**return ultimo == -1;**

**}**

**bool Lista::llena(){**

**return ultimo == TAMLISTA - 1;**

**}**

**void Lista::insertar(int pos, tipo\_dato elem){**

**if (llena() || pos < 0 || pos > ultimo + 1){**

**cout<<"Ingresa un elemento consecutivo valido"<<endl;**

**system("pause");**

**return;**

**}**

**for(int i = ultimo+1 ; i > pos ; i-- ){**

**datos[i] = datos[i - 1];**

**}**

**datos[pos] = elem;**

**ultimo++;**

**}**

**void Lista::elimina(int pos){**

**if (vacia() || pos < 0 || pos > ultimo ){**

**cout<<"La lista esta vacia"<<endl;**

**system("pause");**

**return;**

**}**

**for (int i = pos ; i <= ultimo ; i++){**

**datos[i] = datos[i + 1];**

**}**

**ultimo--;**

**}**

**void Lista::imprimir(){**

**if (vacia()){**

**cout<<"La lista esta vacia"<<endl;**

**system("pause");**

**return;**

**}**

**for(int i = primero ; i <= ultimo ; i++){**

**cout<<"Posicion numero: "<<i<<"\n Dato: "<<datos[i]<< " "<<endl;**

**}**

**}**

**void Lista::recupera(int pos){**

**if (vacia() || pos < 0 || pos > ultimo ){**

**cout<<"La lista esta vacia"<<endl;**

**system("pause");**

**return;**

**}**

**cout<<"Posicion numero: "<<pos<<"\n Dato: "<<datos[pos]<<" "<<endl;**

**cout<<endl;**

**}**

**bool Lista::lista\_ordenada(){**

**int j = primero, i = ultimo;**

**while (j < i){**

**if (datos[j] > datos[j+1]){**

**return 1;**

**}**

**j++;**

**}**

**i--;**

**return 0**

**;**

**}**

**void Lista::localiza(tipo\_dato dato){**

**int opc;**

**if (vacia()){**

**cout<<"La lista esta vacia"<<endl;**

**system("pause");**

**return;**

**}**

**system("cls");**

**cout<<"Que metodo quieres usar :"<<endl;**

**cout<<"1) Metodo lineal"<<endl;**

**cout<<"2) metodo binario"<<endl;**

**cout<<"2) Seleccione opcion : "<<endl;**

**cin>>opc;**

**switch(opc){**

**case 1:{**

**cout<<"\n Metodo lineal\n\n"<<endl;**

**int i;**

**bool encontrado;**

**for ( i = primero ; i <= ultimo ; i++ ){**

**if (datos[i] == dato){**

**cout<<" Posicion numero: "<<i<<"\n Dato: "<<datos[i]<<" "<<endl;**

**cout<<endl;**

**encontrado = true;**

**}**

**}**

**if (encontrado == false){**

**cout<<" Dato no encontrado: "<<endl;**

**}**

**}break;**

**case 2:{**

**if (lista\_ordenada()){**

**cout<<" La lista no esta ordenada"<<endl;**

**return;**

**}**

**int i = 0, j = ultimo, m;**

**bool encontrado;**

**while ( i <= j ){**

**m = ( i + j )/2;**

**if ( datos[m] == dato ){**

**cout<<" Posicion numero: "<<m<<"\n Dato: "<<datos[m]<<" "<<endl;**

**cout<<endl;**

**encontrado = true;**

**}**

**if ( dato < datos[m] ){**

**j = m - 1;**

**}**

**else{**

**i = m + 1;**

**}**

**}**

**if (encontrado == false){**

**cout<<" Dato no encontrado: "<<endl;**

**}**

**}break;**

**}**

**}**

**void Lista::anular(){**

**ultimo = -1;**

**}**

**void Lista::burbuja\_mejorada(){**

**if (vacia()){**

**cout<<"La lista esta vacia"<<endl;**

**system("pause");**

**return;**

**}**

**int i = ultimo, j;**

**int aux;**

**bool cambio;**

**do{**

**cambio = false;**

**j = 0;**

**while (j < i){**

**if (datos[j] > datos[j+1]){**

**aux = datos[j];**

**datos[j] = datos[j+1];**

**datos[j+1] = aux;**

**cambio = true;**

**}**

**j++;**

**}**

**i--;**

**}while(cambio);**

**}**

**void Lista::insercion(){**

**int i = 1, j;**

**int aux;**

**while(i <= ultimo){**

**aux = datos[i];**

**j = i;**

**while( j > 0 && aux < datos[j-1]){**

**datos[j] = datos[j-1];**

**j--;**

**}**

**if(i != j){**

**datos[j] = aux;**

**}**

**i++;**

**}**

**}**

**void Lista::seleccion(){**

**int i = 0, j, m;**

**int aux;**

**while( i < ultimo ){**

**m = i;**

**j = i +1;**

**while( j <= ultimo ){**

**if( datos[j] < datos[m] ){**

**m = j;**

**}**

**j++;**

**}**

**if( m!= i ){**

**aux = datos[i];**

**datos[i] = datos[m];**

**datos[m] = aux;**

**}**

**i++;**

**}**

**}**

**void Lista::mezcla( int primero, int ultimo ){**

**if ( primero >= ultimo ){**

**return;**

**}**

**int medio = ((primero + ultimo)/2);**

**mezcla ( primero, medio );**

**mezcla ( medio + 1, ultimo );**

**for ( int c = primero ; c <= ultimo ; c++ ){**

**arreglo\_copia[c] = datos[c];**

**}**

**int i = primero, j = medio + 1, x = primero;**

**while ( i <= medio && j <= ultimo ){**

**while ( i <= medio && arreglo\_copia[i] <= arreglo\_copia[j] ){**

**datos[x++] = arreglo\_copia[i++];**

**}**

**if ( i <= medio ){**

**while ( j <= ultimo && arreglo\_copia[j] <= arreglo\_copia[i] ){**

**datos[x++] = arreglo\_copia[j++];**

**}**

**}**

**}**

**while ( i <= medio ){**

**datos[x++] = arreglo\_copia[i++];**

**}**

**while ( j <= ultimo ){**

**datos[x++] = arreglo\_copia[j++];**

**}**

**}**

**void Lista::quicksort( int primero, int ultimo ){**

**int aux, j, i;**

**if ( primero >= ultimo ){**

**return;**

**}**

**aux = datos[(primero + ultimo)/2];**

**datos[(primero + ultimo)/2] = datos[ultimo];**

**datos[ultimo] = aux;**

**i = primero, j = ultimo;**

**while ( i < j ){**

**while ( i < j && datos[i] <= datos[ultimo] ){**

**i++;**

**}**

**while ( i < j && datos[j] >= datos[ultimo] ){**

**j--;**

**}**

**if ( i != j ){**

**aux = datos[i];**

**datos[i] = datos[j];**

**datos[j] = aux;**

**}**

**}**

**if ( i != ultimo ){**

**aux = datos[i];**

**datos[i] = datos[ultimo];**

**datos[ultimo] = aux;**

**}**

**quicksort( primero ,i-1 );**

**quicksort( i+1 ,ultimo );**

**}**

**struct Lista mi\_lista;**

**int main(){**

**int opc=0;**

**do{**

**system("cls");**

**menu();**

**cout<<"Continuar 1 salir 12:"<<endl;cin>>opc;**

**}while(opc!=12);**

**system("pause>>cls");**

**return 0;**

**}**

**void menu(){**

**int opc=0;**

**int dato,pos;**

**cout<<" Practica 3"<<endl;**

**cout<<"1) Insertar elemento (Por posicion)"<<endl;**

**cout<<"2) Eliminar elemento"<<endl;**

**cout<<"3) recupera elemento"<<endl;**

**cout<<"4) localiza posicion"<<endl;**

**cout<<"5) Imprimir Lista"<<endl;**

**cout<<"6) Anular Lista"<<endl;**

**cout<<"7) Ordenamiento\_Burbuja"<<endl;**

**cout<<"8) Ordenamiento\_Insercion"<<endl;**

**cout<<"9) Ordenamiento\_Seleccion"<<endl;**

**cout<<"10) Ordenamiento\_Mezcla"<<endl;**

**cout<<"11) Ordenamiento\_Quicksort"<<endl;**

**cout<<"12) Salir"<<endl;**

**cout<<"Seleccione opcion:"<<endl;**

**cin>>opc;**

**switch(opc){**

**case 1: {**

**cout<<"En que posicion desea insertar el elemento :"<<endl;cin>>pos;**

**cout<<"Inserte elemento: "<<endl;cin>>dato;**

**mi\_lista.insertar(pos,dato);**

**}break;**

**case 2: {**

**cout<<"Que posicion deseas eliminar :"<<endl;cin>>pos;**

**mi\_lista.elimina(pos);**

**}break;**

**case 3: {**

**cout<<"Que posicion deseas consultar :"<<endl;cin>>pos;**

**mi\_lista.recupera(pos);**

**}break;**

**case 4: {**

**cout<<"Que dato deseas buscar :"<<endl;cin>>dato;**

**mi\_lista.localiza(dato);**

**}break;**

**case 5: {**

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

**}break;**

**case 6: {**

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

**}break;**

**case 7: {**

**mi\_lista.burbuja\_mejorada();**

**}break;**

**case 8: {**

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

**}break;**

**case 9: {**

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

**}break;**

**case 10: {**

**mi\_lista.mezcla(mi\_lista.primero, mi\_lista.ultimo);**

**}break;**

**case 11: {**

**mi\_lista.quicksort(mi\_lista.primero, mi\_lista.ultimo);**

**}break;**

**case 12:break;**

**default:**

**cout<<"La opcion: "<<opc<<"No existe"<<endl;**

**}**

**}**