Instituto tecnológico de Orizaba

Estructura De Datos

Unidad 1: Introducción a las estructuras de datos

Carrera: Ingeniería en Sistemas Computacionales

Grupo: 3g2B

Hora: 17:00 – 18:00 hrs

Profesora: Martínez Castillo María Jacinta

Alumno: Moran De la Cruz Aziel

Introducción

Las estructuras de datos son herramientas fundamentales en el campo de la informática y la programación. Son formas organizadas y eficientes de almacenar, administrar y manipular datos en un programa o sistema. Una estructura de datos define cómo se organiza la información y cómo se puede acceder y modificar.

Competencia Específica

La eficiencia en tiempo y espacio. Cada estructura de datos tiene diferentes características en términos de acceso, inserción, eliminación y búsqueda de elementos. Al elegir una estructura de datos adecuada para un problema específico, es importante considerar cómo afectará el rendimiento del programa en términos de tiempo de ejecución y consumo de memoria.

Por ejemplo, algunas estructuras de datos son más eficientes en operaciones de acceso rápido, lo que significa que pueden recuperar elementos en tiempos constantes, como los arreglos. Otros, como las listas enlazadas, pueden ser más eficientes en operaciones de inserción y eliminación, ya que no requieren reorganizar los elementos contiguos en memoria.

Marco Teórico

1.1 Clasificación de la estructura de datos

Las dos clases principales de estructuras de datos son las lineales y las no lineales, dependiendo de la complejidad de las relaciones lógicas que representan. Las estructuras de datos lineales incluyen pilas, colas y listas ligadas lineales. Las estructuras de datos no lineales incluyen grafos y árboles.

1.2 Tipos de datos abstractos (TDA)

Un Tipo de dato abstracto (en adelante TDA) es un conjunto de datos u objetos al cual se le asocian operaciones. El TDA proporciona una interfaz con la cual es posible realizar las operaciones permitidas, abstrayéndose de la manera en cómo están implementadas dichas operaciones. Esto quiere decir que un mismo TDA puede ser implementado utilizando distintas estructuras de datos y proporcionar la misma funcionalidad.

El paradigma de orientación a permite el encapsulamiento de los datos y las operaciones mediante la definición de clases e interfaces, lo cual permite ocultar la manera en cómo ha sido implementado el TDA y solo permite el acceso a los datos a través de las fórmulas provistas por la interfaz.

1.3 Ejemplos de TDA´S

Algunos ejemplos del uso de TDA en programación son: Listas: una colección secuencial de elementos con operaciones como agregar, borrar, recorrer. Pilas: una colección de elementos que sigue la propiedad LIFO. Operaciones comunes incluyen push, pop, top.

1.4 Manejo de memoria

Todas las variables, arreglos y objetos en general tienen una duración determinada en el transcurso de un programa. Son creados y destruidos para su uso y después para que la memoria sea liberada para que la utilicen otros objetos.

En Java existen tres formas de usar la memoria para almacenar valores, son:

a) Memoria estática: Es la utilizada por variables globales y las declaradas de tipo static. Estos objetos tienen enfocado la misma dirección de memoria desde el comienzo hasta el final del programa. (variables globales, tipo static)

b) Memoria automática: Es la utilizada por argumentos en una función y por las variables locales. Cada entrada en la función crea estos objetos y son destruidos al salir de ella. (Variables locales, argumentos en una función).

c) Memoria Dinámica: Es también llamado almacenamiento libre porque en este caso el programador es el que solicita memoria para crear los objetos y es el responsable de liberar la memoria cuando ya no la necesita para ser reutilizada.

Materiales y equipo

Los materiales utilizados fueron

Computadora

Netbeans

Material de apoyo para la práctica

Desarrollo

Métodos de datos desordenados

1.- El método validaEspacio comprueba si el arreglo este vacío, si el valor de j es

igual a -1 el arreglo está vacío y sino el arreglo contine algo.

2.- El método agregarDatos si el valor de j es menor o igual al valor del arreglo se podrá ingresar un valor al arreglo sino mostrará que el arreglo está lleno y no se podrá agregar un dato.

3.- El método de imprimirArray con un ciclo for se ejecuta mientras es menor y o igual a j y después el arreglo en la variable que se crea de tipo string se concatenan los diferentes datos guardados y al acabar de concatenarlos se imprimirán

4.- El método busquedaSecuencial, lo primero que hace es pedir un valor donde que se buscara en el arreglo y después se habrá una condicional donde el valor de i debe ser menor o igual a j y también el dato debe de ser diferente a uno de los del arreglo y después si el valor de i es mayor a j retornara le valor de -1 y sino regresara la posición de donde se encuentra el número.

5.- El método de eliminarDato, pide la posición del dato que se quiere eliminar y después se mete es dato en un ciclo for y se asigna a k y se va incrementando ese valor y mientras eso pasa se le asigna a datos[k] el valor de datos[k+1], al terminar eso se decrementa la j.

6.- Creamos una clase nueva que llamaremos “MenuDesordenado”, en dicha clase será donde pongamos nuestro menú para así poder seleccionar lo que deseemos hacer.

Resultados

Al ejecutar el menú nos salen las opciones a las que podremos acceder por medio de botones.

1.- “Agregar datos” nos pedirá que ingresemos un valor ya sea entero o el que hayamos asignado con anterioridad.

2.- “Imprimir” nos mostrará todos los datos que hayamos agregado con “Agregar datos” la diferencia que tiene con el de datos ordenados es que los datos guardarán en el orden en que hayan sido agregados.

3.- “Buscar” nos mostrará la posición en la que encuentre el valor que deseemos buscar.

En caso de que el valor no exista mandará un mensaje diciendo que no se encontró el valor.

4.- “Eliminar” llamará al método que se encargará de eliminar el valor que escribamos siempre y cuando esté en el arreglo, si no mandará un mensaje de que el valor no fue encontrado.

6.- “Salir” hace que termine el programa.

Métodos de datos ordenados.

1.- El método de DatosOrdenados es más que nada para indicar el tamaño del array en donde es almacenado los valores a ordenar.

```
public DatosOrdenados(int tam) {
    ordenados = new int[tam];
    p=-1;
}
```

2.- El método de arrayVacío es el mismo que el de método de datos desordenados, aunque en este caso no usamos j si no que usamos p para comparar si está lleno o vacío el array, si p es igual a -1 entonces está vacío, pero si no entonces si tiene algún valor dentro del array.

```
public boolean arrayVacio () {
    return (p==-1);
}
```

3.- El método de imprimirArray lo ocupamos para la impresión final de los valores de manera ordenada, estos valores los metemos en una cadena para que nos salgan los valores en una misma ventana, la diferencia del método desordenado es que en la cadena en vez de usar “Datos” usamos “Ordenados, solo es el cambio de nombre de variable para identificar de qué es el método, si ordenado odesordenado.

```
public void imprimirArray()
    {
        String cad="";
        for(int i=0;i<=p;i++) {
            cad+="["+i+"]"+ordenados[i]+"\n";
        }
        Tools.imprimirMSJE("Datos del array"+"\n"+cad);
    }
```

4.- El método de busSecuencialOrdenada es similar al método de BusquedaSecuencial desordenada, solo que en este retorna el dato buscado de una manera más eficiente ya que aquí no usamos if y else.

```
public byte busSecuencialOrdenada(int dato) {
    byte i=0;
    while (i<=p && ordenados[i]<dato)
        i++;
    return (byte) ((i>p || ordenados[i]>dato)?-i:i);

}
```

5.- El método de eliminarDato al igual que el de números desordenados sirve para eliminar un dato específico, en este caso solo cambia la variable a ordenados.

```
public void eliminarDato(byte pos) {
        for(int k=pos;k<=p;k++)
        {
            ordenados[k]=ordenados[k+1];
        }
        p--;
    }
```

6.- El método de recorrePosición es el que ocupamos para ordenar los valores agregados ya sea de menor a mayor o de mayor a menos, sea el caso de cómo decidamos indicarlo, este método compara cada uno de los valores y si uno es menor que el otro entonces ese se posicionaría primero, y así con todos los valores.

```
public void recorrePosicion(int pos) {
    for(int j=p+1; j>pos;j--) {
        ordenados[j]=ordenados[j-1];
    }
}
```

7.- El método de agregarOrdenado este es el método que básicamente nos sirve para agregar datos al array, a diferencia del método desordenado en este se hace una comparación para evitar agregar números que ya existen en el array, validar si aún el array tiene espacio.

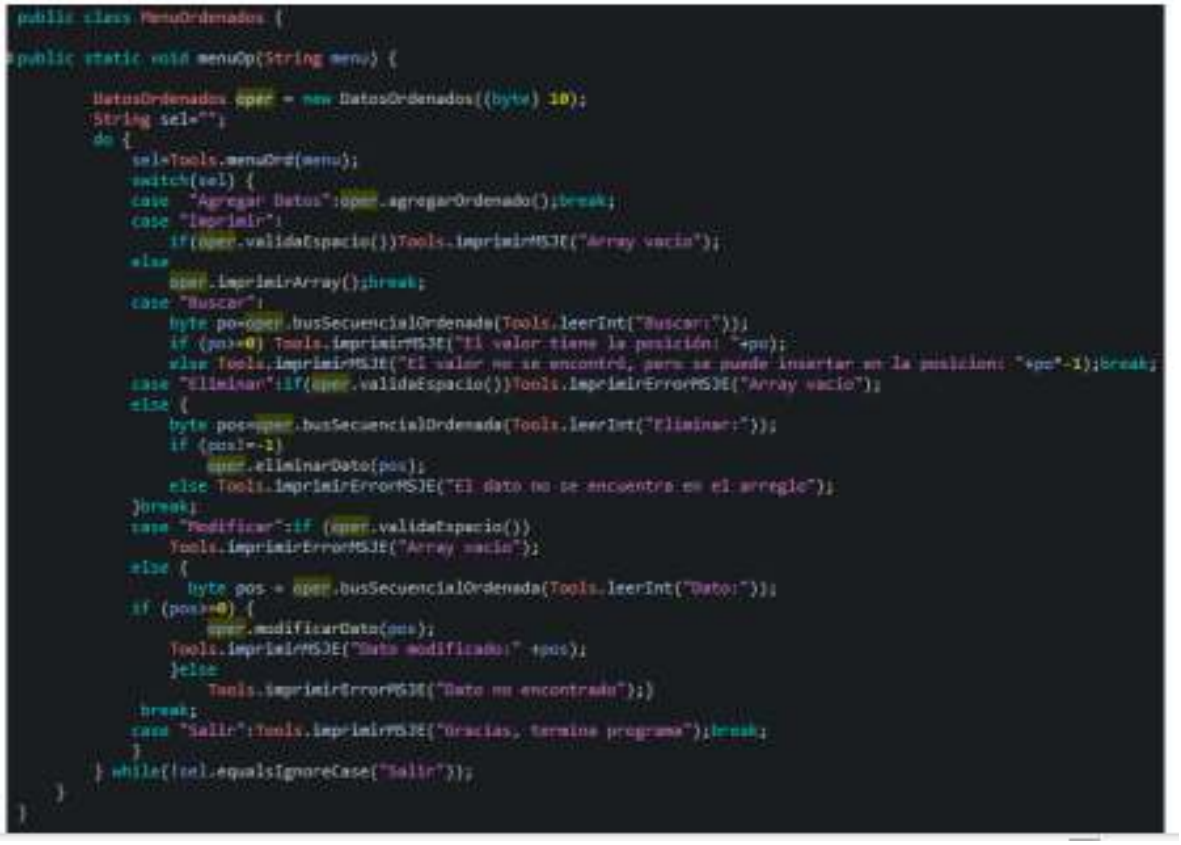
```
public void agregarOrdenado() {
    int dato=Tools.leerInt("Escribe un entero: ");
    if(validaEspacio()) {
        ordenados[p+1]=dato;
        p++;
    } else {
        int pos=busSecuencialOrdenada(dato);
        if(pos>0)Tools.imprimirMSJE("Ya existe el dato");
        else {
            pos*=-1;
            recorrePosicion(pos);ordenados[pos]=dato;
            p++;
        }
    }
}
```

8.- El método de modificarDato cómo dice su nombre es para modificar cierto valor ya existente en el array, este método consta de tres parte uno donde modifica el extremo inferior del array en donde se modificar la posición 0 siempre y cuando sea menor al valor de la posición 1, la segunda parte es del extremo superior en donde solo se modifica mientras sea mayor que el número anterior a su posición, y la tercera parte es para modificar valores centrales, en donde dicho valor a modificar tiene que ser mayor que la posición anterior y menor que la posición que le sigue.

```
public void modificarDato(byte pos) {
    int dato;
    if (pos==0) {
        do {
            dato=Tools.leerInt("Modificar: <:" +ordenados[pos+1]); }
    while (dato>ordenados[pos+1]);
        ordenados[pos]=dato;
    }
    else {
        if (pos==p) {
            do {
                dato=Tools.leerInt("Modificar >:" +ordenados[pos-1]); }
        while (dato<ordenados[pos-1]);
            ordenados[pos]=dato;
        }
        else {
            if (pos>=1 || pos<=p-1) {
                do {
                    dato=Tools.leerInt("Modificar: >:" +ordenados[pos-1]+" y <:" +ordenados[pos+1]); }
            while (dato>ordenados[pos+1] || dato<ordenados[pos-1]);
                ordenados[pos]=dato;
            }

        }
    }
}
```

9.- Creamos una clase nueva que llamaremos “MenuOrdenados”, en dicha clase será donde pongamos nuestro menú para así poder seleccionar lo que deseemos hacer.



```
public class MenuOrdenados {

public static void menuOp(String menu) {

        DatosOrdenados oper = new DatosOrdenados((byte) 10);
        String sel="";
        do {
            sel=Tools.menuOrd(menu);
            switch(sel) {
            case  "Agregar Datos":oper.agregarOrdenado();break;
            case "Imprimir":
                if(oper.validaEspacio())Tools.imprimirMSJE("Array vacio");
            else
                oper.imprimirArray();break;
            case "Buscar":
                byte po=oper.busSecuencialOrdenada(Tools.leerInt("Buscar:"));
                if (po>=0) Tools.imprimirMSJE("El valor tiene la posición: "+po);
                else Tools.imprimirMSJE("El valor no se encontró, pero se puede insertar en la posicion: "+po*-1);break;
            case "Eliminar":if(oper.validaEspacio())Tools.imprimirErrorMSJE("Array vacio");
            else {
                byte pos=oper.busSecuencialOrdenada(Tools.leerInt("Eliminar:"));
                if (pos!=-1)
                    oper.eliminarDato(pos);
                else Tools.imprimirErrorMSJE("El dato no se encuentra en el arreglo");
            }break;
            case "Modificar":if (oper.validaEspacio())
                Tools.imprimirErrorMSJE("Array vacio");
            else {
                  byte pos = oper.busSecuencialOrdenada(Tools.leerInt("Dato:"));
            if (pos>=0) {
                    oper.modificarDato(pos);
                Tools.imprimirMSJE("Dato modificado:" +pos);
                }else
                    Tools.imprimirErrorMSJE("Dato no encontrado");}
             break;
            case "Salir":Tools.imprimirMSJE("Gracias, termine programa");break;
            }
        } while(!sel.equalsIgnoreCase("Salir"));
    }
}
```

Resultados

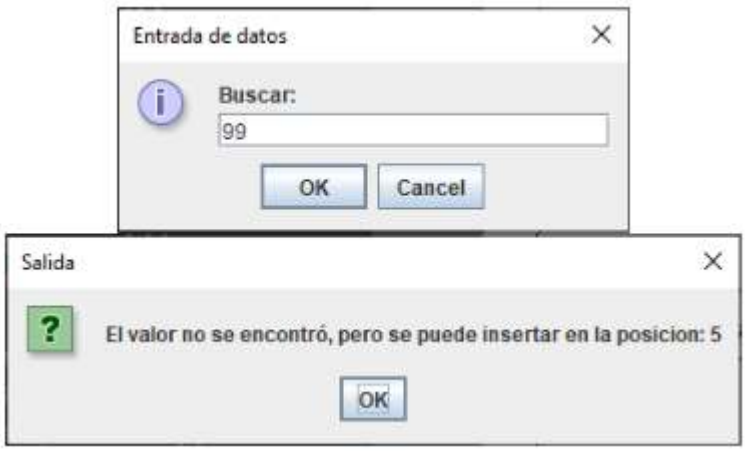
Al ejecutar el menú nos salen las opciones a las que podremos acceder por medio de botones.

1.- “Agregar datos” nos pedirá que ingresemos un valor ya sea entero o el que hayamos asignado con anterioridad.

2.- “Imprimir” nos mostrará todos los datos que hayamos agregado con “Agregar datos” la diferencia que tiene con el de datos desordenados es que los datos se ordenarán de menor a mayor.

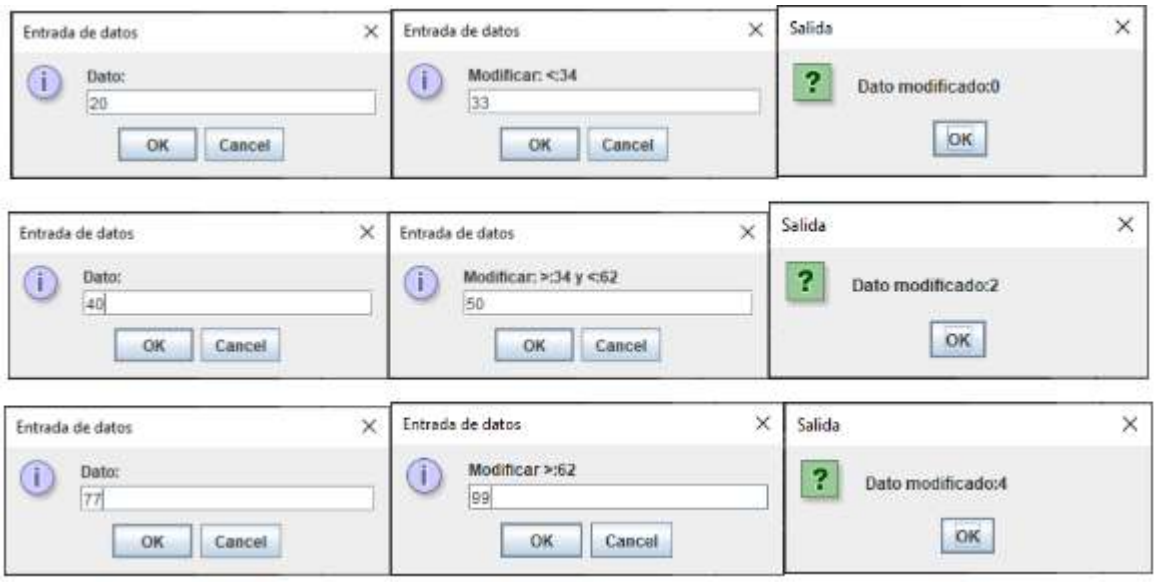
3.- “Buscar” nos mostrará la posición en la que encuentre el valor que deseemos buscar.

En caso de que el valor no exista mandará un mensaje diciendo que no se encontró el valor y a su vez sugerirá que puede agregarse dicho número buscado en la siguiente posición.



4.- “Eliminar” llamará al método que se encargará de eliminar el valor que escribamos siempre y cuando esté en el arreglo, si no mandará un mensaje de que el valor no fue encontrado.

5.- “Modificar” nos permitirá modificar los valores ya ingresados con anterioridad siempre y cuando sea menor que el valor que le sigue y mayor que el valor anterior.



6.- “Salir” hace que termine el programa.

Conclusión

En conclusión, las estructuras de datos son elementos fundamentales en la programación y la informática en general. Estas estructuras nos permiten organizar y manipular eficientemente grandes cantidades de información, lo que resulta clave para el desarrollo de aplicaciones y sistemas complejos. La elección adecuada de la estructura de datos es crucial para lograr un buen rendimiento y eficiencia en los algoritmos utilizados.