

Materia:

ESTRUCTURA DE DATOS

Docente:

María Jacinta Martínez Castillo

Integrantes:

Ríos Méndez Cesar

Sentís Robles Pedro

Hernández Acevedo Brayan

Hora:

17:00 - 18:00

Fecha:

17/02/2023



REPORTE DE PRACTICA.

	Unidad: 1
ENERO-JU NIO 2023	NO. DE PRÁCTICA: 1 NOMBRE DE PRÁCTICA: INTRODUCCION A LA ESTRUTURA DE DATOS.
	DOCENTE: María Jacinta Martínez Castillo
NO. DE CONTROL Y NOMBRE: RIOS MENDEZ CESAR 21011029 HERNANDEZ ACEVEDO BRAYAN 21010962 SENTIES ROBLES PEDRO 21011047	
FECHA ENTREGA: 17/04/202 3	

INTRODUCCIÓN

Es una forma de organizar los ítems en términos de memoria, y también la forma de acceder a cada ítem a través de una lógica definida. Algunos ejemplos de estructuras de datos son pilas, colas, listas enlazadas, árbol binario y muchos más.

Las estructuras de datos realizan algunas operaciones especiales solo como inserción, eliminación y recorrido. Por ejemplo, debe almacenar datos para muchos empleados

donde cada empleado tiene su nombre, identificación de empleado y un número de teléfono móvil. Entonces, este tipo de datos requiere una administración de datos compleja, lo que significa que requiere una estructura de datos compuesta por múltiples tipos de datos primitivos. Por tanto, las estructuras de datos son uno de los aspectos más importantes a la hora de implementar conceptos de codificación en aplicaciones del mundo real.

OBJETIVO GENERAL

Dar al estudiante las bases para que desarrolle y opere estructuras de datos.

OBJETIVO ESPECIFICO

Las estructuras de datos son útiles porque nos permiten tener una batería de herramientas para solucionar ciertos tipos de problemas.

Además, nos permiten hacer un software más eficiente optimizando recursos, algo muy útil para IoT y para los entornos que trabajan con Big Data.

Y para llegar a todo el alumno tendrá que resolver diferentes ejercicios dados por su docente, esto para apoyar su conocimiento y reforzarlo.

MARCO TEÓRICO

1.1 Clasificación de las estructuras de datos

Cuando pensamos en estructuras de datos, generalmente hay cuatro formas: Lineal: arreglos, listas. Árbol: binario, montones, partición de espacio, etc. Hash: tabla hash distribuida, árbol hash, etc.

- Arrays.
- Listas enlazadas.
- Pilas.
- Colas.
- Árboles binarios.

1.2 Tipos de datos abstractos (TDA)

En ciencias de la computación un tipo de dato abstracto (TDA) o tipo abstracto de datos (TAD) es un modelo matemático compuesto por una colección de operaciones definidas sobre un conjunto de datos para el modelo.

Un tipo de dato abstracto o ADT (Abstract Data Type) es definido por una especificación abstracta, es decir, permite especificar las propiedades lógicas y funcionales de un tipo de dato.

1.3 Ejemplos de TDA's

Pilas

Colas

Listas

Autómata finito

Tipos de datos

Clase

Matriz, Vector

1.4 Manejo de memoria

Memoria Estática

La forma más fácil de almacenar el contenido de una variable en memoria en tiempo de ejecución es en memoria estática o permanente a lo largo de toda la ejecución del programa.

Memoria Dinámica

Supongamos que nuestro programa debe manipular estructuras de datos de longitud desconocida. Un ejemplo simple podría ser el de un programa que lee las líneas de un archivo y las ordena.

Reserva de memoria.

La función `g_malloc` posibilita la reserva de una zona de memoria, con un número de bytes que le pasemos como parámetro. Además, también existe una función similar llamada `g_malloc0` que, no sólo reserva una zona de memoria, sino que, además, llena esa zona de memoria con ceros, lo cual nos puede beneficiar si se necesita un zona de memoria totalmente limpia.

Liberación de memoria.

Cuando se hace una reserva de memoria con `g_malloc` y, en un momento dado, el uso de esa memoria no tiene sentido, es el momento de liberar esa memoria. Y el sustituto de `free` es `g_free` que, básicamente, funciona igual que la anteriormente mencionada.

Realojamiento de memoria

En determinadas ocasiones, sobre todo cuando se utilizan estructuras de datos dinámicas, es necesario ajustar el tamaño de una zona de memoria (ya sea para hacerla más grande o más pequeña).

Asignación dinámica

El proceso de compactación del punto anterior es una instancia particular del problema de asignación de memoria dinámica, el cual es el cómo satisfacer una necesidad de tamaño n con una lista de huecos libres. Existen muchas soluciones para el problema.

MATERIAL



• Equipo de cómputo
• Eclipse (u otro software de programación)

SECRETARÍA DE EDUCACIÓN PÚBLICA



TECNOLÓGICO
NACIONAL DE MÉXICO



METODOLOGIA

Emplear un dato, cuyo valor cambia, implica usar una variable, que en programación imperativa equivale a darle un nombre a un lugar de la memoria, donde se almacenará el valor que se necesite. Esto en contrapartida con el uso de constantes que son justamente valores que no cambian. Hasta ahora hemos hablado de almacenar en una variable un único valor simple, de un tipo dado, a la vez. Sin embargo, más de una vez será necesario conservar más de un valor.

Esto es fácil de realizar empleando variables para cada uno de los valores que sean necesarios.

Sin embargo, hay veces que estos datos, por alguna razón, tienen una unidad conceptual y conviene mantenerlos juntos y no separados en variables distintas.

Por ejemplo, todos los datos que corresponden a una persona, tales como su edad, el estado civil, su nombre y apellido, dirección, documento de identidad, etc. Así, estos datos, deberían, entonces, poder ser agrupados en una unidad, que llamaremos estructura de datos. Una estructura de datos es un conjunto de datos que se agrupan por alguna razón. Este agrupamiento lleva implícito un almacenamiento de cada uno de los componentes de la estructura.

Llamaremos elementos de la estructura a cada uno de sus componentes, es decir, a cada uno de los datos que forman parte de ella.

Una estructura de datos presupone un conjunto de elementos y tiene, al menos potencialmente, la posibilidad de tener más de un elemento, de lo contrario no podría hablarse de estructura.

OPERACIONES

Distinguiremos tres operaciones fundamentales que pueden realizarse sobre una estructura de datos y sus elementos:

COLOCAR un elemento nuevo en la estructura. Para el caso de las estructuras estáticas, como esto implica que la misma no puede crecer, hablaremos entonces de asignar un valor a un elemento de la estructura, es decir, cambiar un valor por otro.

Para el caso de las estructuras dinámicas esto implica que la cantidad de elementos de la misma aumenta.

Hablaremos, entonces, de una inserción.

SACAR, de la estructura, un elemento ya existente en ella. Para el caso de las estructuras estáticas esto implica que la misma no disminuirá su tamaño. Hablaremos, en este caso, de una asignación que se hará sobre un valor ya existente, haciendo que éste último cambie su valor. Para el caso de las estructuras dinámicas esto implica que la cantidad de elementos de la misma disminuye.

Hablaremos, entonces, de una supresión.

INSPECCIONAR un elemento de la estructura, para conocer su valor. Si bien la inspección se hace de a un elemento por vez, es decir, no puede hacerse simultáneamente con más de un elemento, sí puede hacerse, inspeccionando de a uno por vez, de forma sistemática sobre algunos o todos los elementos de la estructura. La inspección de los elementos NO modifica el valor contenido en los mismos ni el tamaño de las estructuras. Se dice que la inspección (o lectura del valor contenido en un elemento) no es destructiva. En realidad, para las estructuras estáticas no existe una diferencia notable entre las operaciones de COLOCAR y SACAR, simplemente se trata de cambiar un valor de un elemento por otro valor, es decir, de llevar a cabo una asignación de un nuevo valor.

DESARROLLO

Para esta practica que realizaremos se llevaran los siguientes pasos:

ARRAY OPERACIONES.

En esta parte haremos métodos que son de una pila los cuales nos ayudaran a entender el entorno que conforma la pila.

```
public class ArrayOperacion {  
    private int datos[];  
    private byte j;  
    public ArrayOperacion(int tam){  
        datos = new int [tam];  
        j =-1;  
    }  
}
```

```
}
```

En este caso creamos las variables que es el contador *J* y el arreglo de *datos* y después creamos el constructor para inicializar las variables.

```
public boolean ArrayVacio(){  
    return(j==-1);  
}
```

1	2	3	4	5	6	7
---	---	---	---	---	---	---

En este método lo que se estará haciendo es comprobar si hay datos en el arreglo recorriéndolo si es que no hay datos retornara que el arreglo esta vacío.

```
public void AgregarDatos(){  
    if(j<datos.length-1){  
        datos[j+1] = Tools.leeEntero("Escribe un valor: ");  
        j++;  
    } else{  
        JOptionPane.showMessageDialog(null, "Array lleno.");  
    }  
}
```

Se llama al método. E insertamos un dato cada que lo llamemos.

0	1	2	3	4
---	---	---	---	---

Se vuelve a llamar al método.

0	1	2	3	4
---	---	---	---	---

Se vuelve a llamar al método.

0	1	2	3	4
---	---	---	---	---

Y así hasta llenar el arreglo.

En este método lo que aremos será ir pidiendo los datos y irlos agregando al arreglo claramente esto mientras se siga pidiendo el método y terminará cuando la persona llegue a la capacidad del arreglo o al número deseado.

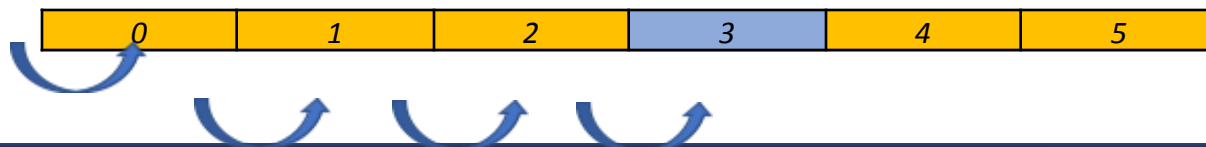
```
public void imprimirArray(){
String cad = "";
for (int i = 0; i <= j; i++) {
cad += "[" + i + "]" + " : " + datos[i] + "\n";
}
JOptionPane.showMessageDialog(null, "Datos del Array:\n" + cad);
}
```

ImprimirArray: el método como su nombre lo dice va a imprimir los datos del arreglo haciendo recorridos y concatenando los resultados para poder imprimir lo que haya dentro del arreglo.

```
public byte BuscarSecuencial(int dato){
byte i =0;
while(i<=j && dato!= datos[i])
i++;
if(i<=j){
return i;
}
else{
return (-1);
}
}
```

Por ejemplo: queremos buscar el dato que esta en la posición 3 el método hace un recorrido

Hasta llegar ala posición donde se encuentra el dato a buscar y retorna la posición.



El método de búsqueda secuencial nos ayuda a encontrar datos en el arreglo para ayudarnos a modificar o eliminar datos por eso este método también esta acompañado de los métodos modificar y eliminar.

```
public void EliminarDato(byte Pos){
for (int k = Pos; k <= j; k++) {
datos[k] = datos[k+1];
}
```



```

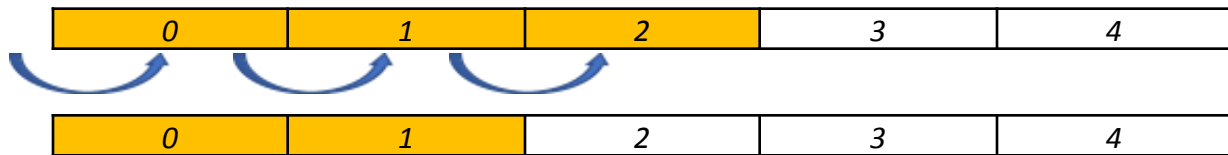
}
j--;
}

```

Ejemplo: queremos eliminar un elemento con el método. Para esto ya tenemos un arreglo previamente llenado.



Hace un recorrido hasta llegar al primer elemento para después eliminarlo.



El método eliminar lo que ara será eliminar el primer elemento de la lista recorriendo claramente primero el arreglo, este método solo ira eliminando los primeros elementos de la lista.

```

public void ModificarDato(byte pos){
byte a= Tools.leeByte("Ingrese un nuevo numero: ");
datos[pos]= a;
}

```

El método nos ayudara a modificar una posición, pero también para poder usar bien el método es necesario acompañarlo del método eliminar y búsqueda secuencial.

MENU

El menú es donde se correrá el programa y donde aremos las interacciones con los distintos métodos como se muestra a continuación.

```

public class uno {
public static void main(String[]args) {
String menu= "Insertar dato,Imprimir,Buscar Dato, SALIR ";
Examen oper=new Examen(5);
String sel;
{
do{
sel=Tools.seleccionBoton8(menu);
switch(sel) {

```



EDUCACIÓN



TECNOLÓGICO



```

case "Insertar dato": oper.agregarDatos();break;// Insertar datos
case "Busqueda Secuencial": oper.busquedaSecuencial(); break;//Buscar dato
case "Imprimir datos":oper.imprimirArray();//Imprimir los datos del arreglo
if (oper.arrayVacio())Tools.imprimirErrorMSJE("Array vacio");
else //si no hay datos imprimirá que el arreglo esta bacio si tiene datos
imprimirá el arreglo

oper.imprimirArray();

break;

case "eliminar":if(oper.arrayVacio())Tools.imprimirErrorMSJE("Array vacio");
else {
byte pos=oper.busquedaSecuencial();
if(pos!=-1)
oper.eliminarDato(pos);
else Tools.imprimirErrorMSJE("El dato no se encuentra en el arreglo");
}

case "salir": Tools.imprimirErrorMSJE("fin del programa");
}

}while(!"salir".equals(sel));
}
}

```

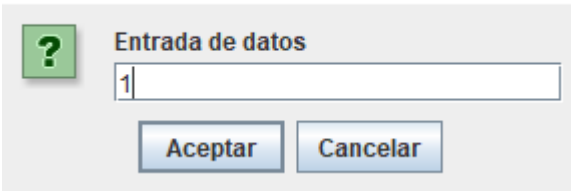
CAPTURAS PARA EVIDENCIA.

Ventana emergente.



Método insertar dato.

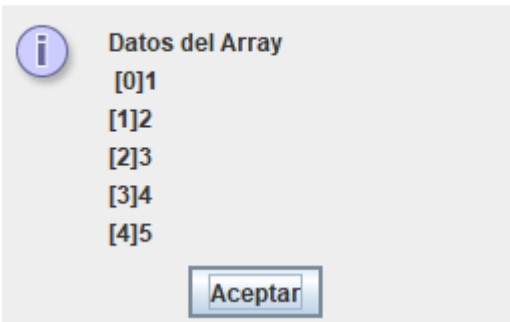
Escribe un valor: X



A dialog box titled "Entrada de datos" with a green question mark icon. It contains a text input field with the value "1" and two buttons: "Aceptar" and "Cancelar".

Imprimir datos.

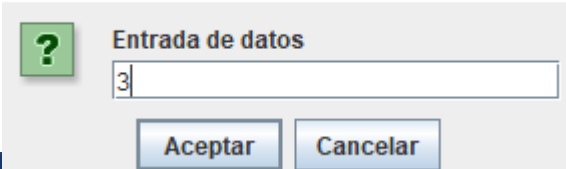
Mensaje X



A dialog box titled "Datos del Array" with an information icon. It displays the following text:
[0]1
[1]2
[2]3
[3]4
[4]5
At the bottom is an "Aceptar" button.

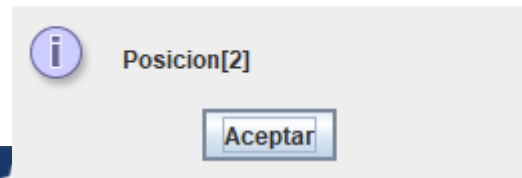
Búsqueda secuencial.

Escribe un valor: X



A dialog box titled "Entrada de datos" with a green question mark icon. It contains a text input field with the value "3" and two buttons: "Aceptar" and "Cancelar".

Mensaje X



A dialog box titled "Posicion[2]" with an information icon. It contains an "Aceptar" button.



EDUCACION
SECRETARÍA DE EDUCACIÓN PÚBLICA

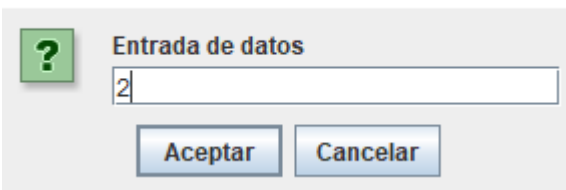


TECNOLÓGICO
NACIONAL DE MÉXICO



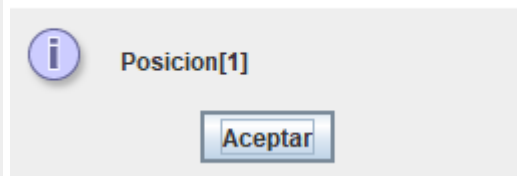
Eliminar dato.

Escribe un valor: X



A dialog box titled "Entrada de datos" with a green question mark icon. It contains a text input field with the value "2" and two buttons: "Aceptar" and "Cancelar".

Mensaje X



A dialog box titled "Posicion[1]" with an information icon. It contains an "Aceptar" button.

DATOS ORDENADOS.

En esta clase lo que cambia son algunos métodos los cuales ayudan a ordenar los datos mientras se van insertando en el arreglo.

```
public class DatosOrdenados {
    private int ordenados[];
    private byte p;

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

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

    public String imprimeDatos(){
        String cad = " ";
        for (int i = 0; i <= p; i++) {
            cad += i + "["+ordenados [i]+"]" + "\n";
        }
        return "\n"+cad;
    }

    public byte busSecuenciaOrdenada (int dato)
    {
        byte i=0;
        while (i<ordenados.length && ordenados[i]<dato)
            i++;
        if (i<ordenados.length || ordenados[i]>dato)
            return (byte) -i;
        else
            return i;
    }

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

```

}
p--;
}
public void recorrePosiciones(byte pos) {
for(int j=p+1; j>pos; j--) {
ordenados[j]=ordenados[j-1];
}
}
public void agregarOrdenados() {
int dato=Tools.LeerInt("Escribe un entero");
if(validaVacio()) {
ordenados[p+1]=dato;
p++;
}else {
byte pos=busSecuenciaOrdenada(dato);
if (pos>0)Tools.imprimirMSJE("Ya existe el dato");
else{
pos*=-1;
recorrePosiciones((byte)pos);ordenados[pos]=dato; p++;
}
}
}
}

```

CONCLUSION

La introducción hecha por la maestra nos ayudó a entender mejor cómo funciona una pila, el cómo funcionan sus métodos y como se comportan estos, gracias a los ejercicios planteados y propuestos por el docente conseguimos un entendimiento mas amplio del tema además que nos proporciono material aparte con el cual buscaba un apoyo ala hora de enseñar, también resaltar la atención puesta por parte de la docente ala hora de hacer los trabajos ya que nos atendía en nuestros problemas o dificultades que presentábamos.

Concluimos bajo nuestro punto de vista que lo enseñado y estudiado durante el tema fue bien estructurado para llegar a entender del tema y posterior mente cuando empezamos a realizar los ejercicios nos llenamos de mas conocimientos y habilidades las cuales fuimos mejorando con forme pasaban los ejercicios.

BIBLIOGRAFIA

Henry, R. (2022). ¿Qué es una estructura de datos en programación? | Henry. Henry.
<https://blog.soyhenry.com/que-es-una-estructura-de-datos-en-programacion/#:~:text=En%20el%20%C3%A1mbito%20de%20la,correcta%20para%20un%20determinado%20problema>

Sena, M. (2021, 7 diciembre). Estructuras de Datos - TechWo - Medium. Medium.
<https://medium.com/techwomenc/estructuras-de-datos-a29062de5483>

colaboradores de Wikipedia. (2023). Estructura de datos. Wikipedia, la enciclopedia libre.
https://es.wikipedia.org/wiki/Estructura_de_datos

Montero, L. (2018, 4 julio). Pilas (Stacks) - Laboratoria Devs - Medium. Medium.
<https://medium.com/laboratoria-developers/pilas-stacks-76f3b8a31f61>