Tecnológico Nacional de México

Campus Orizaba

ESTRUCTURA DE DATOS

Carrera:

Ingeniería en sistemas computacionales

Tema 1: Introducción a las estructuras de datos

Alumnos:

Tezoco Cruz Pedro-20011328

Flores Domínguez Ángel Gabriel-21010951

Grupo:3g2B

Fecha 17/02/23

Introducción

La clasificación de las estructuras de datos es un tema fundamental en la informática. Se refiere a la forma en que se organizan y almacenan los datos en un programa.

Los tipos de datos abstractos (TDA) son una forma de encapsular los datos y las operaciones que se realizan con ellos. Un TDA define un conjunto de operaciones y restricciones sobre un conjunto de datos y proporciona una interfaz para acceder a esos datos. En este reporte de dan algunos ejemplos de TDA cada uno tiene un conjunto de operaciones y restricciones específicas que definen su comportamiento y uso.

El manejo de memoria es otro tema importante en la informática, y se refiere a cómo se gestiona y utiliza la memoria de un sistema informático. La memoria puede ser estática o dinámica. La memoria estática se reserva durante el tiempo de compilación y se utiliza para almacenar variables y estructuras de datos que no cambian en el tiempo de ejecución. La memoria dinámica se asigna durante el tiempo de ejecución y se utiliza para almacenar estructuras de datos que cambian en el tiempo de ejecución.

Competencia Específica

Conoce y comprende las diferentes estructuras de datos, su clasificación y forma de manipularlas para buscar la manera más eficiente de resolver problemas.

Marco Teórico:

1.1. Clasificación de las estructuras de datos:

Las estructuras de datos se pueden clasificar en varios tipos, como lineales y no lineales, estáticas y dinámicas, homogéneas y heterogéneas, entre otros. A continuación, describiré cada uno de estos tipos con más detalle:

Lineales: son aquellas estructuras de datos en las que los elementos están organizados en una secuencia lineal, es decir, uno después de otro. Algunos ejemplos de estructuras lineales son las listas, las pilas, las colas y las matrices.

No lineales: son aquellas estructuras de datos en las que los elementos no están organizados en una secuencia lineal. Algunos ejemplos de estructuras no lineales son los árboles, los grafos y las estructuras jerárquicas.

Estáticas: son aquellas estructuras de datos cuyo tamaño se define en tiempo de compilación y no puede cambiar durante la ejecución del programa. Algunos ejemplos de estructuras estáticas son las matrices y los arrays.

Dinámicas: son aquellas estructuras de datos cuyo tamaño se define en tiempo de ejecución y puede cambiar durante la ejecución del programa. Algunos ejemplos de estructuras dinámicas son las listas enlazadas y los árboles.

Homogéneas: son aquellas estructuras de datos en las que todos los elementos tienen el mismo tipo de dato. Un ejemplo de estructura homogénea es la matriz.

Heterogéneas: son aquellas estructuras de datos en las que los elementos pueden tener diferentes tipos de dato. Un ejemplo de estructura heterogénea es la estructura de datos struct en C.

1.2. Tipos de datos abstractos:

Un tipo de dato abstracto (TDA) es una abstracción matemática que define un conjunto de valores y operaciones sobre ellos. Los TDAs se definen por sus operaciones, no por su implementación, lo que permite que los desarrolladores de software utilicen las operaciones definidas para los TDAs sin preocuparse por la implementación subyacente. Algunos ejemplos de TDAs son las listas, las pilas, las colas, los árboles, los conjuntos y los mapas.

1.3. Ejemplos de TDA:

Continuando con la explicación de los TDAs, a continuación, te proporcionaré algunos ejemplos adicionales de estructuras de datos abstractas:

Conjunto ordenado: es un conjunto que tiene un orden definido. Los elementos se pueden agregar y eliminar, y se pueden buscar por valor o posición.

Pila con prioridad: es una pila en la que los elementos tienen una prioridad asociada. Los elementos se eliminan de la pila en función de su prioridad.

Cola de prioridad: es una cola en la que los elementos tienen una prioridad asociada. Los elementos se eliminan de la cola en función de su prioridad.

Árbol binario de búsqueda: es un árbol en el que cada nodo tiene un valor y dos hijos, llamados hijo izquierdo y hijo derecho. Los valores de los nodos izquierdos son menores que el valor del nodo padre, mientras que los valores de los nodos derechos son mayores.

Grafo dirigido: es un conjunto de nodos conectados por aristas direccionales. Los nodos pueden representar cualquier cosa y las aristas pueden tener un peso o una dirección.

Mapa ordenado: es un mapa en el que las claves están ordenadas. Los elementos se pueden agregar y eliminar, y se pueden buscar por clave o posición.

1.4. Manejo de memoria:

El manejo de memoria es el proceso de asignar y liberar memoria en un programa. La memoria se puede asignar de dos maneras: estática y dinámica.

Memoria estática: se refiere a la memoria que se asigna durante la compilación y no cambia durante la ejecución del programa. Los ejemplos de memoria estática son las variables globales y las variables estáticas.

Memoria dinámica: se refiere a la memoria que se asigna durante la ejecución del programa y puede cambiar en tiempo de ejecución.

El manejo de memoria es importante porque permite a los programadores asignar la cantidad necesaria de memoria para sus programas sin desperdiciar recursos.

1.4.1. Memoria estática operaciones básicas:

Las operaciones básicas que se pueden realizar con memoria estática son la declaración de variables y la asignación de valores a esas variables.

Declaración de variables: es el proceso de definir una variable en el código fuente. La sintaxis de la declaración de variables varía según el lenguaje de programación utilizado.

Asignación de valores: es el proceso de asignar un valor a una variable. La sintaxis de la asignación de valores también varía según el lenguaje de programación utilizado.

En resumen, el manejo de memoria y la comprensión de los TDAs son fundamentales para el desarrollo de programas eficientes y escalables. Con un buen manejo de memoria y la utilización de los TDAs adecuados, los programadores pueden crear programas que se ejecuten rápidamente y sean fáciles

Materiales:

- Una computadora
- Eclipse IDE
- GitLab

Datos Ordenados

Se crean todos los métodos para llenar un arreglo con datos desordenados.

```
1  package Actualizacion;
2
3  import javax.swing.JOptionPane;
6
7  public class DatosOrdenados {
8     private int ordenados[];
9     private byte p;
0     public DatosOrdenados(int tam) {
1         ordenados=new int [tam];
2         p=-1;
3     }
4
5     public boolean validaVacio() {
6         return(p==-1);
7     }
8
9     public void imprimirArray() {
0          String cad="";
1          for(int i=0;i<=p;i++) {
2              cad+="\n"+"["+i+"]"+ordenados[i]+"\n";
3              }
4
5          Tools.imprimeMsje("Datos del array= "+ cad);
6      }
7
8     public byte busSecuencialOrdenada(int dat) {
9         byte i = 0;
0         while(i <= p && ordenados[i] < dat)
1             i++;
2
3         return (byte) ((i > p || ordenados[i] > dat)?-i:i);
4     }
5
```

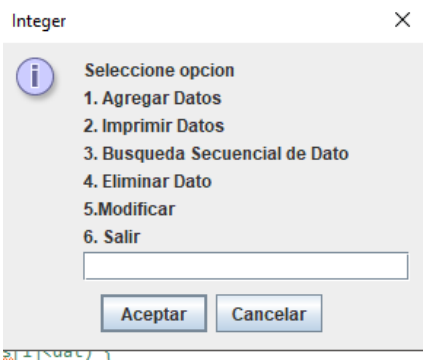
```
36    public void eliminarDato(byte pos) {
37         for(int k=pos;k<p;k++) {
38             ordenados[k]=ordenados[k+1];
39         }
40         p--;
41     }
42
43    public void recorrePosiciones(byte pos) {
44     // este metodo baja todos los elementos para que haya una casilla vacia
45        for (int j = p+1; j > pos; j--) {
46            ordenados[j] = ordenados[j-1];
47
48        }
49
50    }
51    public void agregarOrdenado(){
52        int dato = Tools.leerInteger("Ingrese Dato:");
53        if(validaVacio ()){
54            ordenados[p+1] = dato;
55            p++;
56
57        } else {
58            byte pos = busSecuencialOrdenada(dato);
59            //System.out.println(pos);
60
61            if(pos > 0){
62               JOptionPane.showMessageDialog(null, "El dato ya existe.");
63            } else{
64                pos*=-1;
65                recorrePosiciones(pos);
66                ordenados[pos] = dato;
67                p++;
68            }
69        }
70    }
71
```

```
public void modificaOrde(byte pos)   {
    int dato=Tools.leerInteger("Cual es el nuevo dato=");
    if (pos==0 && dato < ordenados[pos+1]) {
        ordenados[pos]=dato;
    }
    else {
        if(pos==p && dato> ordenados[pos-1]) {
            ordenados[pos]=dato;
        }
        else {
            if(pos>0 && pos<p) {
                ordenados[pos]=dato;
            }
            else {
                Tools.imprimeMsje("No fue posible modificar");
            }
        }
    }
}
```

MENU de datos ordenados

```java
package Actualizacion;

import EntradaSalida.Tools;

public class testOrdenados {

    public static void menuOrdenado(String menu) {
        DatosOrdenados oper= new DatosOrdenados(7);
        int sel;
        do {
            sel=Tools.leerInteger(menu+"Seleccione opcion"+"\n"
        +"1. Agregar Datos"+"\n"+
         "2. Imprimir Datos"+"\n"+
        "3. Busqueda Secuencial de Dato"+"\n"
         +"4. Eliminar Dato"+"\n"
        +"5.Modificar"+"\n"+
         "6. Salir");

        switch(sel) {
               case 1:oper.agregarOrdenado();
               break;
               case 2:if(oper.validaVacio()) {
                   Tools.imprimeMsje("El arreglo esta vacio.");break;
               }else
                   oper.imprimirArray();
               break;
               case 3:int b=Tools.leerInt("Dato para Buscar:");
                   if(oper.busSecuencialOrdenada(b)<0)
               Tools.imprimeMsje("El dato no se encontro");
               else
                   Tools.imprimeMsje("El dato se encontro en la pos: "+oper.busSecuencialOrdenada(b));
               break;
               case 4: byte datoBuscad = Tools.leerByte("Ingresa el Dato a Eliminar: ");
               byte resultadoBusqued = oper.busSecuencialOrdenada(datoBuscad);
               if(resultadoBusqued >-1){
                   JOptionPane.showMessageDialog(null, "Número encontrado en la posición: [" + resultadoBusqued + "]",
                           "Busqueda Secuencial",JOptionPane.INFORMATION_MESSAGE);
                   oper.eliminarDato(resultadoBusqued);
               } else{

                   JOptionPane.showMessageDialog(null, "No se encontro el numero","Busqueda secuencial",JOptionPane.WARNING_MESSAGE);
               }
               break;
               case 5:int po=Tools.leerInt("Dame el dato a modificar");
               byte resu = oper.busSecuencialOrdenada(po);
                   oper.modificaOrde(resu);
       break;
               case 6:Tools.imprimeMsje("Fin del programa");break;
               default:Tools.err("Opcion no definida intenta de nuevo");
        }
        }while(sel!=6);
    }

```

Menú en ejecución:



Array

Se crean lo métodos para agregar datos a un array

```java
package Metodos;

import EntradaSalida.Tools;

public class ArrayOperacion {
    //Dos atributos
    private int datos[];
    private byte J;

    public ArrayOperacion(int tam) {//dimensiono
        datos= new int[tam];
        J=-1;
    }

    public boolean ArrayVacio() {//Valida si el arreglo esta vacio
    return(J==-1);
    }

    public void agregarDatos() {//no se recibe nada

        if(J<datos.length-1) {
            datos[J+1]=Tools.leerInteger("Escribe un valor:");
            J++;
        }
        else
            Tools.err("Array Lleno");//mensaje de error
    }

    public byte buscarSecuencial(int dato) { // Este metodo en el case

        byte i=0;
        while(i<=J && dato!=datos[i]) {
            i++;}
        if(i>J)

            return(-1);

        else
            return i;
```

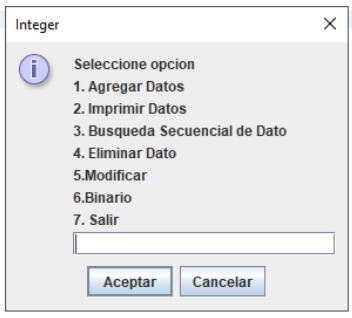
```
42      public void elimina() {}
43
44      public void eliminarDato(byte pos) {
45          for(int k=pos;k<J;k++) {
46              datos[k]=datos[k+1];
47          }
48          J--;
49      }
50
51      public void modificar(byte pos) {
52          byte y=Tools.leerByte("Ingrese nuevo numero: ");
53          datos[pos]=y;
54              }
55
56
57
58
59      public void imprimirArray() {
60          String cad="";
61          for(int i=0;i<=J;i++) {
62              cad+="\n"+"["+i+"]"+datos[i]+"\n";
63              }
64
65          Tools.imprimeMsje("Datos del array= "+ cad);
66      }
67      public String decimalBinario(int valor) {
68          String bin="";
69          while (valor!=0) {
70              bin=valor%2+bin;
71              valor/=2;
72          }
73          return bin;
74      }
75
```

```
public void imprimirArrayBnario() {
    String c="";
    for(int i=0;i<=J;i++) {
        c+="\n"+"["+i+"]"+datos[i]+" -Valor binario= "+decimalBinario(datos[i])+"\n";
        }
    Tools.imprimeMsje("Datos del array= "+ c);
    }
```

Menú

```
1  package TestMetodos;
2
3  import EntradaSalida.Tools;
5
6  public class Menu {
7
8
9     public static void menu1(String menu) {
10         ArrayOperacion oper= new ArrayOperacion(5);
11         int sel;
12         do {
13             sel=Tools.leerInteger(menu+"Seleccione opcion"+
14                     "\n"+"1. Agregar Datos"+
15                     "\n"+"2. Imprimir Datos"+
16                     "\n"+"3. Busqueda Secuencial de Dato"+"\n"
17                     +"4. Eliminar Dato"+"\n"+
18                      "5.Modificar"+"\n"+
19                     "6.Binario"+"\n"+
20                      "7. Salir");
21
22         switch(sel) {
23                case 1:oper.agregarDatos();
24                break;
25                case 2:if (oper.ArrayVacio())
26                Tools.imprimeMsje("Array vacio");
27                else
28                oper.imprimirArray();
29                break;
30                case 3:int dato = Tools.leerInt("Elemento que deseas buscar: ");
31                if (oper.buscarSecuencial(dato)==-1) {
32                Tools.imprimeMsje("No se encontro el numero");}
33                else{
34                Tools.imprimeMsje("Se encontro el valor en la posicion: " + oper.buscarSecuencial(dato));}
35                break;
36                case 4:if(oper.ArrayVacio()) {Tools.imprimeMsje("Array Vacio");
37                }else
38                {
38                  {
39                    byte pos = oper.buscarSecuencial(Tools.leerInteger("Dato a eliminar:"));
40                  if (pos!=-1) {
41                      Tools.imprimeMsje("El dato eliminado estaba en la posicion : "+pos);
42                      oper.eliminarDato(pos);
43                  }else Tools.err("El dato no se encuentra en el arreglo");}
44                  break;
45                  case 5:if(oper.ArrayVacio()==false) {
46                  byte u=oper.buscarSecuencial(Tools.leerInteger("Dato a modificar:"));
47                      oper.modificar(u);}
48                  else Tools.imprimeMsje("No hay datos");break;
49                  case 6:oper.imprimirArrayBnario();break;
50                  case 7:Tools.imprimeMsje("Fin del programa");break;
51                  default:Tools.err("Opcion no definida intenta de nuevo");
52           }
53          }while(sel!=7);
54     }
```

Programa en ejecución:



Ejercicios hechos en clase

```java
public class Ejercicios{


    public static void generarValores(int n) {

        double sum=0;
        for(int i=0; i<n;i++) {

            sum+=calcularFactorial(2*i)/calcularFactorial(i);
        }

        System.out.println("Resultados="+sum);
    }

    public static float calcularFactorial(int valor) {

        float aux=1;
        int f=1;
        while(f<=valor) {
            aux=f*aux;
            f++;
        }
        return aux;
    }

    public static boolean validarAmstrong(int dato) {

        int valor=dato,sum=0;
        while(valor>0) {
            sum+=Math.pow((valor%10),3);
        valor/=10;  }
        return(sum==dato);
    }
```

```java

    public static boolean ternaPitagota(int a, int b, int c) {

        int result = (int) (Math.pow(a, 2) + Math.pow(b, 2));
        int valor = (int) (Math.pow(c, 2));
        return (valor == result);

    }


    public static String capicua(int numero) {
        int r = 0;
        int Num = numero;
        while (numero!= 0) {
            int ultimoDig = numero % 10;
            r = (r * 10) + ultimoDig;
            numero /= 10;
        }
        if (Num==r) {
            return "Es capicua";
        }
        return "No es capicua";
    }


    public static int totalDigi(int valor) {
        int c= 0;
        while (valor!=0) {
            valor/=10;
            c++;
        }
        return c;
    }
```

```java
    public static boolean primo(int num) {
        boolean primo= true;
        if (num%2!=0) {
            return primo;
        }
        return primo=false;
    }



    public static String binario(int n) {
        if (n == 0) {
            return "0";
        }
        String binary = "";
        while (n > 0) {
            int rem = n % 2;
            binary = rem + binary;
            n = n / 2;
        }
        return binary;
    }
```

Conclusiones:

En conclusión, existen diversos métodos para el arreglo de datos abstractos que pueden ser utilizados en diferentes situaciones y contextos. Algunos de estos métodos son más adecuados para ciertas tareas que otros, dependiendo de los requisitos específicos del problema a resolver.

Es importante considerar factores como el tiempo de ejecución, el espacio de almacenamiento y la complejidad del código al elegir un método para el arreglo de datos abstractos. También es fundamental tener en cuenta las estructuras de datos disponibles y las operaciones que se necesitan realizar para manipular los datos.

Algunos de los métodos de arreglo de datos abstractos más comunes incluyen la ordenación, búsqueda, filtrado y agrupación. Cada uno de estos métodos tiene sus propias técnicas y algoritmos específicos que se utilizan para manipular los datos de manera eficiente y efectiva.

Bibliografía:

Tipo de Dato Abstracto. (s/f). Google.cNo title. (s/f). Zyrosite.com. Recuperado el 22 de abril de 2023, de https://estructuradedatosbalv.zyrosite.com/about-me-copy-copy-ehjecMOmuA

No title. (s/f). Zyrosite.com. Recuperado el 22 de abril de 2023, de https://estructuradedatosbalv.zyrosite.com/about-me-copy-copy-ehjecMOmuA

Memoria dinamica y memoria estatica - Programacion en el lenguaje C MARG. (s/f). Google.com. Recuperado el 22 de abril de 2023, de https://sites.google.com/site/programacionencmarg/home/6-memoria-dinamica-y-memoria-estatica