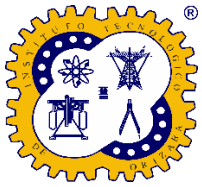


Instituto Tecnológico de Orizaba

CARRERA

ING. INFORMATICA

ASIGNATURA

PROGRAMACION ORIENTADA A OBJETOS

TEMA 5

ARRAY

ALUMN@:

MAYTE MELLADO HUERTA

KEVIN ALAN ORTIZ FLORES

ANGEL DE JESUS CONTRERAS HERRERA

NO.CONTROL

21010202

21010207

21010180

GRUPO

2a3B

FECHA DE ENTREGA

02/06/2023

INTRODUCCION

El tema principal de este reporte es sobre los arreglos en la programación, y se abordan varios subtemas relacionados. Estos subtemas incluyen los arreglos unidimensionales y multidimensionales, que son elementos fundamentales para almacenar y organizar datos de manera eficiente. A lo largo de este informe, exploraremos en detalle cada uno de estos subtemas, comprendiendo su funcionamiento y aplicando los conceptos aprendidos en ejercicios prácticos.

5.1 Unidimensionales:

En esta sección, nos adentraremos en los arreglos unidimensionales, también conocidos como vectores. Estos arreglos nos permiten almacenar una colección lineal de elementos del mismo tipo, los cuales se guardan de forma contigua en la memoria. Aprenderemos a declarar y utilizar arreglos unidimensionales, así como a realizar operaciones comunes como la inserción, eliminación, búsqueda y modificación de elementos. Además, exploraremos conceptos clave como el índice del arreglo y cómo acceder a elementos específicos. Para reforzar estos conceptos, llevaremos a cabo una serie de ejercicios prácticos donde aplicaremos las operaciones mencionadas y resolveremos problemas utilizando arreglos unidimensionales.

5.2 Multidimensionales:

En esta sección, nos sumergiremos en el fascinante mundo de los arreglos multidimensionales. A diferencia de los arreglos unidimensionales, los arreglos multidimensionales nos permiten organizar los datos en estructuras bidimensionales o tridimensionales. Exploraremos cómo declarar y utilizar arreglos multidimensionales, acceder a elementos utilizando índices múltiples y realizar operaciones específicas en cada dimensión. También veremos cómo los arreglos multidimensionales pueden ser útiles en situaciones que requieren el manejo de información más compleja, como matrices y tablas. Pondremos en práctica nuestros

conocimientos a través de ejercicios que involucran arreglos multidimensionales, resolviendo problemas de búsqueda, ordenamiento y manipulación de datos.

En conclusión, este reporte nos ha brindado una visión integral de los arreglos unidimensionales y multidimensionales en la programación. Hemos aprendido a utilizar estas poderosas estructuras de datos para almacenar, acceder y manipular información de manera efectiva. Los arreglos unidimensionales nos permiten manejar secuencias lineales de datos, mientras que los arreglos multidimensionales nos brindan una organización más compleja en forma de matrices y tablas. Mediante ejercicios prácticos, hemos fortalecido nuestras habilidades en la implementación de operaciones básicas y resolución de problemas utilizando arreglos unidimensionales y multidimensionales.

COMPETENCIA ESPECIFICA

Conoce y aplica programas que implementen el uso de arreglos para reconocerlos como una herramienta para agrupar datos y facilitar la solución de problemas.

MARCO TEÓRICO:

5.1 Unidimensional: Los arreglos unidimensionales, también conocidos como vectores, son estructuras de datos fundamentales en programación que nos permiten almacenar y organizar una colección lineal de elementos del mismo tipo. Estos elementos se almacenan de forma contigua en la memoria, y cada elemento es identificado por un índice que representa su posición en el arreglo.

La declaración de un arreglo unidimensional implica especificar su tipo de datos y su tamaño. Una vez creado, podemos acceder a los elementos del arreglo utilizando el índice correspondiente. Es importante destacar que los índices en los arreglos unidimensionales comienzan desde 0 y van hasta el tamaño del arreglo menos 1.

Los arreglos unidimensionales nos permiten realizar diversas operaciones, como la inserción de elementos en una posición específica, la eliminación de elementos, la búsqueda de un valor en el arreglo y la modificación de elementos existentes. Estas operaciones se realizan mediante la manipulación de los índices y la asignación de valores a los elementos del arreglo.

Además, los arreglos unidimensionales son especialmente útiles en situaciones donde se necesita almacenar y procesar grandes cantidades de datos de manera eficiente. Por ejemplo, se utilizan en algoritmos de búsqueda, ordenamiento y procesamiento de datos.

5.2 Multidimensional: Los arreglos multidimensionales son estructuras de datos que nos permiten almacenar y organizar datos en dos o más dimensiones. A diferencia de los arreglos unidimensionales, los arreglos multidimensionales se componen de filas y columnas, formando una estructura en forma de matriz o tabla.

La declaración de un arreglo multidimensional implica especificar su tipo de datos y las dimensiones del arreglo. Por ejemplo, un arreglo bidimensional tiene dos dimensiones: filas y columnas. Cada elemento del arreglo se identifica por un par de índices correspondientes a su posición en la matriz.

Los arreglos multidimensionales son útiles en situaciones donde se requiere organizar datos en una estructura tabular, como una matriz de registros o una tabla de doble entrada. Estas estructuras de datos permiten realizar operaciones más complejas, como la búsqueda y modificación de elementos en diferentes dimensiones, así como el cálculo de sumas, promedios y otros análisis en datos organizados.

Además, los arreglos multidimensionales pueden ser utilizados en algoritmos que involucran problemas espaciales o de simulación, como el procesamiento de imágenes, el análisis de redes sociales y la resolución de problemas en ciencias e ingeniería.

MATERIAL Y EQUIPO:

- ♥ Computadora
- ♥ NetBeans IDE 8.3
- ♥ PDF y presentaciones dadas por el docente

DESARROLLO DE LA PRACTICA

5.1 Unidimensional:

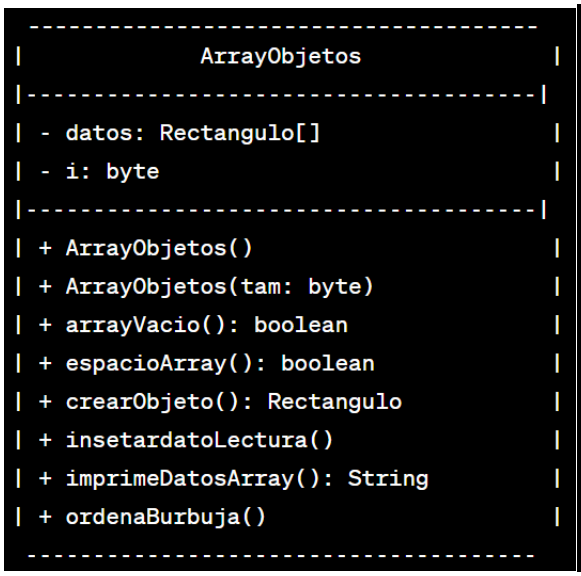
1. Declaración y creación de un arreglo unidimensional: Comienza por declarar e inicializar un arreglo unidimensional en tu lenguaje de programación preferido. Asegúrate de especificar el tipo de datos y la longitud del arreglo.
2. Acceso a los elementos del arreglo: Utiliza los índices adecuados para acceder a los elementos individuales del arreglo. Puedes utilizar bucles para recorrer el arreglo y realizar operaciones en cada elemento.
3. Inserción y eliminación de elementos: Implementa métodos o funciones que permitan insertar y eliminar elementos en el arreglo. Asegúrate de ajustar la longitud del arreglo y reorganizar los elementos según sea necesario.
4. Búsqueda de elementos: Implementa algoritmos de búsqueda para encontrar elementos específicos en el arreglo. Puedes utilizar técnicas como la búsqueda lineal o la búsqueda binaria, dependiendo de las características del arreglo y de los requisitos de búsqueda.
5. Modificación de elementos: Crea métodos o funciones que permitan modificar los valores de los elementos existentes en el arreglo. Puedes proporcionar opciones para actualizar un elemento en una posición específica o modificar varios elementos en función de ciertos criterios.
6. Algoritmos de ordenamiento: Explora diferentes algoritmos de ordenamiento, como el ordenamiento por inserción, el ordenamiento por selección o el ordenamiento de burbuja. Implementa estos algoritmos en métodos separados y utilízalos para ordenar los elementos del arreglo.

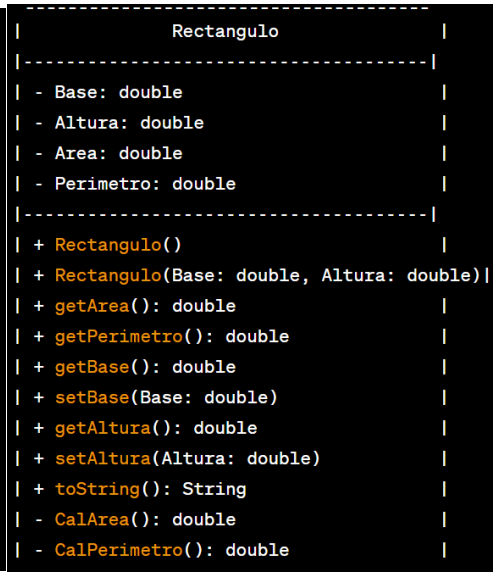
5.2 Multidimensional:

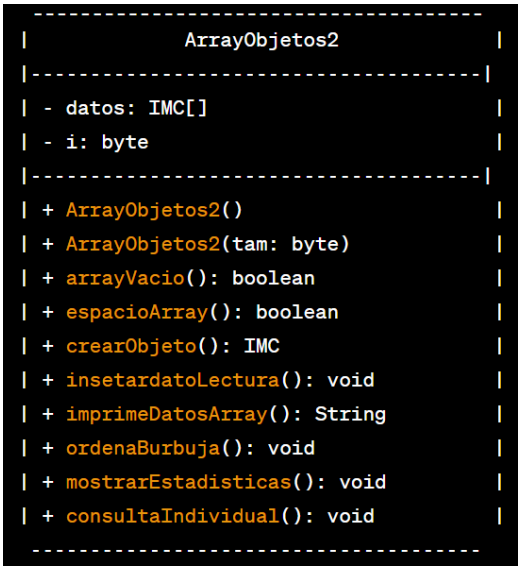
1. Declaración y creación de un arreglo multidimensional: Declara e inicializa un arreglo multidimensional, como una matriz o una tabla, en tu lenguaje de programación. Especifica el número de dimensiones y la longitud de cada dimensión.
2. Acceso a los elementos del arreglo: Utiliza los índices correspondientes a las filas y columnas para acceder a los elementos individuales del arreglo. Puedes utilizar bucles anidados para recorrer el arreglo multidimensional y realizar operaciones en cada elemento.
3. Inserción y eliminación de elementos: Implementa métodos o funciones que permitan insertar y eliminar elementos en el arreglo multidimensional. Asegúrate de ajustar las dimensiones del arreglo y reorganizar los elementos según sea necesario.
4. Búsqueda de elementos: Implementa algoritmos de búsqueda adaptados a arreglos multidimensionales. Puedes utilizar técnicas como la búsqueda lineal o la búsqueda binaria en función de las características del arreglo y de los requisitos de búsqueda.
5. Modificación de elementos: Crea métodos o funciones que permitan modificar los valores de los elementos existentes en el arreglo multidimensional. Proporciona opciones para actualizar un elemento en una posición específica o modificar varios elementos en función de ciertos criterios.
6. Algoritmos de ordenamiento: Explora diferentes algoritmos de ordenamiento adaptados a arreglos multidimensionales. Implementa estos algoritmos en métodos separados y utilízalos para ordenar los elementos del arreglo en diferentes dimensiones.

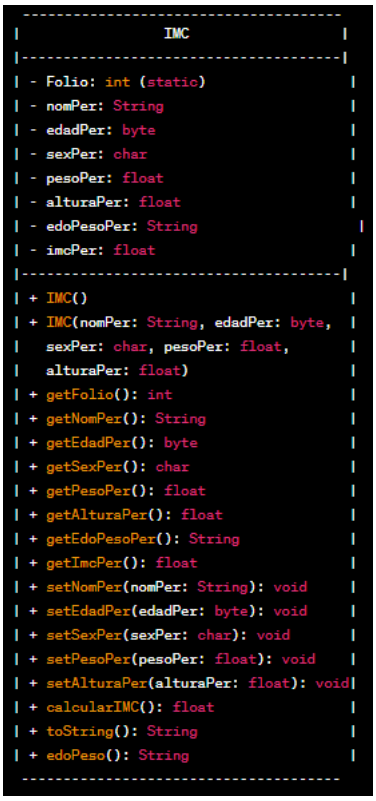
A continuación, se presentarán los diagramas de clase de todos los programas realizados:

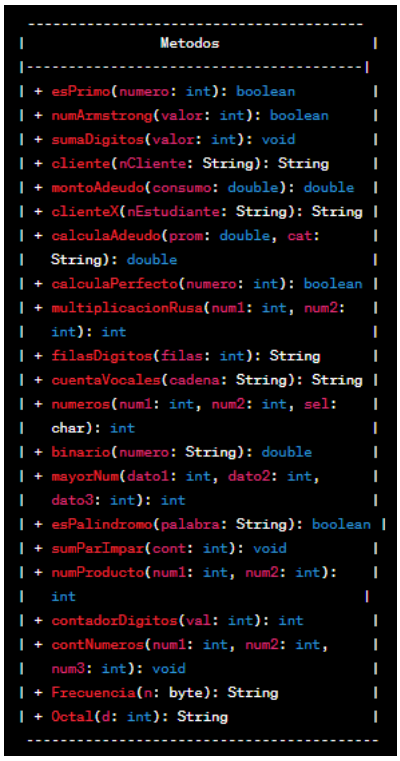

AlmacenarDatos
- datos: Object[]
- i: byte
+ AlmacenarDatos(max: int)
+ arrayVacio(): boolean
+ arrayEspacio(): boolean
+ insetardatoLectura(dato: Object)
+ imprimeDatosArray(): String
+ imprimePrimos(): String
+ convertirABinario(): int
+ imprimeFrecuencia(): String
+ imprimeTriangulo(): String
+ imprimesumaDigitos(): String


ArrayObjetos
- datos: Rectangulo[]
- i: byte
+ ArrayObjetos()
+ ArrayObjetos(tam: byte)
+ arrayVacio(): boolean
+ espacioArray(): boolean
+ crearObjeto(): Rectangulo
+ insetardatoLectura()
+ imprimeDatosArray(): String
+ ordenaBurbuja()


Rectangulo
- Base: double
- Altura: double
- Area: double
- Perimetro: double
+ Rectangulo()
+ Rectangulo(Base: double, Altura: double)
+ getArea(): double
+ getPerimetro(): double
+ getBase(): double
+ setBase(Base: double)
+ getAltura(): double
+ setAltura(Altura: double)
+ toString(): String
- CalArea(): double
- CalPerimetro(): double


ArrayObjetos2
- datos: IMC[]
- i: byte
+ ArrayObjetos2()
+ ArrayObjetos2(tam: byte)
+ arrayVacio(): boolean
+ espacioArray(): boolean
+ crearObjeto(): IMC
+ insetardatoLectura(): void
+ imprimeDatosArray(): String
+ ordenaBurbuja(): void
+ mostrarEstadisticas(): void
+ consultaIndividual(): void


IMC
- Folio: int (static)
- nomPer: String
- edadPer: byte
- sexPer: char
- pesoPer: float
- alturaPer: float
- edoPesoPer: String
- imcPer: float
+ IMC()
+ IMC(nomPer: String, edadPer: byte, sexPer: char, pesoPer: float, alturaPer: float)
+ getFolio(): int
+ getNomPer(): String
+ getEdadPer(): byte
+ getSexPer(): char
+ getPesoPer(): float
+ getAlturaPer(): float
+ getEdoPesoPer(): String
+ getImcPer(): float
+ setNomPer(nomPer: String): void
+ setEdadPer(edadPer: byte): void
+ setSexPer(sexPer: char): void
+ setPesoPer(pesoPer: float): void
+ setAlturaPer(alturaPer: float): void
+ calcularIMC(): float
+ toString(): String
+ edoPeso(): String


Metodos
+ esPrimo(numero: int): boolean
+ numArmstrong(valor: int): boolean
+ sumaDigitos(valor: int): void
+ cliente(nCliente: String): String
+ montoAdeudo(consumo: double): double
+ clienteX(nEstudiante: String): String
+ calculaAdeudo(prom: double, cat: String): double
+ calculaPerfecto(numero: int): boolean
+ multiplicacionRusa(num1: int, num2: int): int
+ filasDigitos(filas: int): String
+ cuentaVocales(cadena: String): String
+ numeros(num1: int, num2: int, sel: char): int
+ binario(numero: String): double
+ mayorNum(dato1: int, dato2: int, dato3: int): int
+ esPalindromo(palabra: String): boolean
+ sumParImpar(cont: int): void
+ numProducto(num1: int, num2: int): int
+ contadorDigitos(val: int): int
+ contNumeros(num1: int, num2: int, num3: int): void
+ Frecuencia(n: byte): String
+ Octal(d: int): String

RESULTADOS

```java
package ArrayBidimen;

import ArrayUnidimen.Metodos;
import EntradaSalida.Tools;

public class AlamacenarDatosTabla {
    private Object matriz[][];
    private byte i,j;//i filas y j columnas
    int matriz2[][]={{18,-3,14},{35,4,20},{4,-16,70},{24,3,10}};

    public AlamacenarDatosTabla(byte m,byte n){
        matriz= new Object[m][n];
        i=-1;j=-1;

    }
    public boolean matrizVacia(){
        return(i==-1 && j==-1);
    }
    public boolean espacioMatriz(){
        return(i<matriz.length && j<matriz[0].length);
    }
    public void leerFilas(){
        byte k,l=0;
        if(espacioMatriz()){

            for( k=(byte)(i+1);k<matriz.length;k++){
                for( l=(byte)(j+1);l<matriz[0].length;l++){
                    matriz[k][l]=Tools.leerInt("Dato a insertar");
                }
            }
            i=k;j=l;
        }
    }
    public String verMatriz(){

    String cad="";
        for(byte k=0;k<matriz.length;k++){
                for(byte l=0;l<matriz[0].length;l++){
                    cad+="[ "+matriz[k][l]+" ]";
                }
                cad+="\n";
    }
        return cad;

    }
```

```
46      public int sumaDiagPrinc (){
47      int suma=0;
        for(int i=0;i<matriz.length;i++){
49          suma+=(int)matriz[i][i];
50      }
51      return suma;
52      }
53
54      public String verMatrizOctal(){
55
56      String cad="";
57          for(byte k=0;k<matriz.length;k++){
58                  for(byte l=0;l<matriz[0].length;l++){
59                      cad+="[ "+matriz[k][l]+" ] "+" ( "
60                              +Metodos.Octal((int) matriz[k][l])+" ) ";
61                  }
62                  cad+="\n";
63      }
64          return cad;
65      }
66
67      public String ordenaBurbuja(){
68          int q = matriz2.length;
69          int n =matriz2[0].length;
            for (int i=0;i<q;i++)
71              for (int h=0;h<n;h++)
72                  for (int g=0;g<4;g++)
73                      for (int f=0;f<n;f++){
74                          if(matriz2[i][h]>=matriz2[g][f])
                                matriz2[i][h] = matriz2[i][h];
76                          else {
77                              int temp = matriz2[g][f];
78                              matriz2[g][f]= matriz2[i][h];
79                              matriz2[i][h] = temp;
80                          }
81                      }
82         String resultado = "";
            for (int i = 0; i < q; i++) {
84          for (int h = 0; h < n; h++) {
85              resultado += matriz2[i][h] + " ";
86          }
87          resultado += "\n";
88      }
89
90      return resultado;
91  }
```

```
92
93      public String verMatriz(int a){
94
95          String cad="";
96          for(byte k=0;k<matriz2.length;k++){
97              for(byte l=0;l<matriz2[0].length;l++){
98                  cad+="[ "+matriz2[k][l]+" ]";
99              }
100             cad+="\n";
101         }
102         return cad;
103     }
104     public String imprimirTriangularInferior() {
105     int tamaño = matriz.length;
106     String trianInfe = "";
107
        for (int i = 0; i < tamaño; i++) {
            for (int j = 0; j <= i; j++) {
110             trianInfe += (matriz[i][j] + " ");
111         }
112         trianInfe += "\n";
113     }
114     return trianInfe;
115 }
117 public String verMatrizTrianInfe() {
118     int tamaño = matriz.length;
119     String cad = "";
120
        for (int i = 0; i < tamaño; i++) {
            for (int j = 0; j < tamaño; j++) {
123             if (j <= i) {
124                 cad += (matriz[i][j] + " ");
125             } else {
126                 cad += "   ";
127             }
128         }
129         cad += "\n";
130     }
131     return cad;
132 }
133 }
```

Esta clase proporciona funcionalidades relacionadas con el almacenamiento, manipulación y visualización de datos en una matriz bidimensional.

```java
package TestPrueba;

import ArrayBidimen.AlamacenarDatosTabla;
import EntradaSalida.Tools;
import java.util.Random;

public class TestAlmacenarDatosTabla {
    public static void main(String [] args){

       menuArrayMatriz("Aleatorio,Leer,Suma Principal,Octal,Constantes,Ordenar,"
                + "Triangulo Inferior,Imprimir,Salir");


    }
    public static void menuArrayMatriz(String menu){
        AlamacenarDatosTabla mat=new AlamacenarDatosTabla((byte)4,(byte)4);

        int matriz2[][]={{18,-3,14},{35,4,20},{4,-16,70},{24,3,10}};
        int dato;
        String sel= "",cad=" ";

        do{
            sel=Tools.boton(menu);
            switch(sel){
                case "Aleatorio":
                    Random aleatorio=new Random();
                    dato=aleatorio.nextInt(21);

                break;
                case "Leer":
                    if(!mat.espacioMatriz())Tools.salidaError("Matriz llena");
                    else{mat.leerFilas();
                    Tools.imprimePantalla("Datos almacenados:\n"+mat.verMatriz()
                    );}
                break;
                case "Suma Principal":
                    Tools.imprimePantalla("Datos almacenados:\n"
                            +mat.sumaDiagPrinc()+"\n"+mat.verMatriz());
                break;
                case "Octal":
                    Tools.imprimePantalla("Datos almacenados:\n"
                            +mat.verMatrizOctal());
                break;
```

```
                case "Constantes":
                    Tools.imprimePantalla("Datos Almacendos:\n"
                            + mat.verMatriz(1));
                    break;
                case "Ordenar":
                        Tools.imprimePantalla("Datos Almacendos:\n"
                                + mat.ordenaBurbuja()+"\n"+ mat.verMatriz(1));
                    break;
                case "Triangulo Inferior":
                    Tools.imprimePantalla("Datos Almacenados:\n"
                            + mat.imprimirTriangularInferior()+"\n"
                            + mat.verMatrizTrianInfe());
                    break;
                case "Imprimir":
                    if(mat.matrizVacia())
                        Tools.salidaError("Array Vacio..insertar datos");
                    else Tools.imprimePantalla("Datos Almacendos:\n"
                            + mat.verMatriz());
                break;
                case "Salir": break;

            }
        }while(!sel.equalsIgnoreCase("Salir"));
    }
}
```

Este código proporciona una interfaz de usuario para interactuar con la clase AlmacenarDatosTabla y probar sus funcionalidades.

```java
package ArrayUnidimen;

import EntradaSalida.Tools;

public class Metodos {
    public static boolean esPrimo(int numero)
    {
        if(numero<2){
            return false;
        }
        for(int i=2;i<=Math.sqrt(numero);i++){
            if(numero%i==0){
                return false;
            }
        }
        return true;
    }
    public static boolean numArmstrong(int valor){

        int aux=valor, sum=0;

        while(aux>0){
            sum+=Math.pow(aux%10, 3);
            aux/=10;
        }

        return(sum== valor);
    }
    public static void sumaDigitos(int valor)
        {
                int suma=0;
                while(valor!=0)
                {
                        suma+=valor%10;
                        valor/=10;
                }
                Tools.imprimePantalla("Suma de digitos:"+suma);
        }
    public static String cliente(String nCliente){

        return(nCliente);
    }

    public static double montoAdeudo(double consumo){


```

```
46
47          double adeudo = 0;
48          if(consumo <100)
49              adeudo = consumo * 40;
50          else
51              if (consumo>=100 && consumo<500)
52                  adeudo = (consumo - 100) * 60 + 4000;
53              else
54                  if (consumo>=500 && consumo<=1000)
55                      adeudo = (consumo - 499) * 80 + 33940;
56                  else
57                      if (consumo>1000)
58                          adeudo = (consumo - 1000) * 100 + 113940;
59          if (adeudo > 600)
60              adeudo = (adeudo* .02) + adeudo;
61          return(adeudo);
62      }
63      public static String clienteX(String nEstudiante){
64
65          return(nEstudiante);
66      }
67
68      public static double calculaAdeudo(double prom, String cat){
69          double adeudo = 0;
70          int pago = 0;
71
72          if(cat.equalsIgnoreCase("A"))
73              pago = 1200;
74          else
75              if(cat.equalsIgnoreCase("B"))
76                  pago = 1000;
77              else
78                  if(cat.equalsIgnoreCase("C"))
79                      pago = 900;
80                  else
81                      if(cat.equalsIgnoreCase("D"))
82                          pago = 600;
83          if(prom >80 && prom<=100)
84              adeudo = pago- (pago*.15);
85          else
86              if(prom >75 && prom<=80)
87                      adeudo=pago - (pago*.08);
88
89          return(adeudo);
90      }
```

```java
    public static boolean calculaPerfecto(int numero){

        boolean nPerfecto = false;
        int suma = 0;
        for (int i = 1; i < numero; i++) {
            if (numero % i == 0) {
                suma = suma + i;
            }
        }
        if(suma==numero)
            nPerfecto = true;

        return(nPerfecto);
    }
    public static int multiplicacionRusa(int num1, int num2){
        int multiplicacion=0;
        while(num1!=0){
            if(num1 % 2 != 0){
                multiplicacion = multiplicacion + num2;
            }

            num1 =num1 / 2;
            num2 = num2 * 2;System.out.println(num1+"  "+num2+"\n");
        }
        return multiplicacion;
    }
    public static String filasDigitos(int filas){
        String calculo = "";
        for(int x=1;x<=filas;x++){

            for(int y=0;y<=((x-1)+y);x++){
                if(y%2==1){
                    calculo += y + " ";
                }

            }
            calculo+="\n";

        }
        return calculo;
    }
```

```
132     public static String cuentaVocales(String cadena){
133         byte a=0,e=0,i=0,o=0,u=0;
134         byte f=0;
            String cad="";
136         while(f<cadena.length()){
137             switch(cadena.charAt(f)){
138 
139                 case 'A' :
140                 case 'a' : a++;break;
141                 case 'E' :
142                 case 'e' : e++;break;
143                 case 'I' :
144                 case 'i' : i++;break;
145                 case 'O' :
146                 case 'o' : o++;break;
147                 case 'U' :
148                 case 'u' : u++;break;
149             }
150             f++;
151 
152         }
153         cad="a="+ Tools.imprimeFrecuencia(a)+"\n"+"e="+ Tools.imprimeFrecuencia(
154                 "i="+ Tools.imprimeFrecuencia(i)+"\n"+"o="+ Tools.imprimeFrecuen
155                 "u="+ Tools.imprimeFrecuencia(u)+"\n";
156         return cad;
157 
158     }
159     public static int numeros(int num1,int num2, char sel){
160 
161         int result=0;
162 
163         switch(sel){
164 
165             case '+' : result=num1+num2;break;
166             case '-' : result=num1-num2;break;
167             case '*' : result=num1*num2;break;
168             case '/' : result=num1/num2;break;
169             case '%' : result=num1%num2;break;
170 
171 
172         }
173 
174         return (result);
175 
176     }
```

```
177     public static double binario(String numero){
            int x = 0, digito= 0;
179         double binario=0;
180         int num = Tools.leerInt(numero);
181         while(num!=0){
182             digito = num%2;
183             binario = binario + digito*Math.pow(10, x);
184             x++;
185             num /=2;
186         }
187
188         return binario;
189     }
190     public static int mayorNum(int dato1,int dato2,int dato3){
191         int a=Math.max(dato1,dato2);
192         Tools.imprimePantalla("El mayor : "+(Math.max(a,dato3)));
193         return a;
194     }
195     public static boolean esPalindromo(String palabra)
196     {
197         palabra = palabra.toLowerCase().replace(" ", "").replace(",","");
198         int cont = 0, inverso = palabra.length()-1;
199         boolean palindromo = false;
200
201         while ((cont<inverso) && (!palindromo)){
202             if (palabra.charAt(cont) == palabra.charAt(inverso)){
203                 cont++;
204                 inverso--;
205             } else {
206                 palindromo = true;
207             }
208         }
209         return palindromo;
210     }
211     public static void sumParImpar(int cont)
212     {
213         int i=1, sImpares=0, sPares=0;
214         while(i<=cont){
215             if(i%2==0)
216                 sPares+=i;
217             else
218                 sImpares+=i;
219             i++;
220         }
221         Tools.imprimePantalla("La suma de los numeros pares es de "+ sPares +
222                 " y la suma de los numeros impares es de " + sImpares);
223     }
```

```
224     public static int numProducto(int num1, int num2)
225     {
226         int suma=0;
227         for(int i =0; i<num2;i++)
228             suma+=num1;
229 
230         return suma;
231     }
232     public static int contadorDigitos(int val){
233         int s=0;
234         while(val>0){
235             s++;
236             val/=10;
237         }
238         return s;
239     }
240     public static void contNumeros(int num1, int num2, int num3)
241     {
242         if(num1==num2 && num2==num3)
243             Tools.imprimePantalla("Los tres numeros son iguales");
244         else if(num1==num2 && num3>num1)
245             Tools.imprimePantalla("Los primeros dos numeros son iguales y el num
246         else if(num1==num3 && num2>num1)
247             Tools.imprimePantalla("El primer y tercer numero son iguales y el nu
248         else if(num3==num2 && num1>num2)
249             Tools.imprimePantalla("El segundo y tercer numero son iguales y el n
250         else if (num1>num2 && num2>num3)
251             Tools.imprimePantalla("El numero " + num1 + " es el mayor");
252         else if (num1<num2 && num2>num3)
253             Tools.imprimePantalla("El numero " + num2 + " es el mayor");
254         else if (num1<num2 && num2<num3)
255             Tools.imprimePantalla("El numero " + num3 + " es el mayor");
256     }
257     public static String Frecuencia(byte n){
258         String cad=" ";
259         for(int i=1;i<=n;i++){
260             cad+="*";
261         }
262         return cad;
263 
264     }
265     public static String Octal(int d){
266         String octal=" ";int mod;int div=d;
267         while(div>=1){
268         mod=div%8;
269         div=div/8;
270         octal=mod+octal;
271         }
272                 return octal;
273 
274     }
```

Estos métodos proporcionan diferentes funcionalidades para realizar cálculos y operaciones sobre los datos pasados como argumentos.

```java
//Arreglos homogeneos
package ArrayUnidimen;

import ArrayUnidimen.Metodos;
import EntradaSalida.Tools;

public class AlmacenarDatos {
    private Object datos[];
    private byte i;//subindice

    public AlmacenarDatos(int max){
        datos=new Object[max];
        i=-1;//para indicar que el array esta vacio
    }
    public boolean arrayVacio(){
        return(i==-1);
    }
    public boolean arrayEspacio(){
        return(i<datos.length-1);
    }
    public void insetardatoLectura(Object dato){
        if(arrayEspacio())
        {
            i++;
            datos[i]=dato;
```

```
26          }else
27              Tools.salidaError("Array Lleno");
28          }
29      public String imprimeDatosArray(){
30
31          String cad=" ";
32          for(byte j=0;j<=i;j++){
33              cad+=j+"["+datos[j]+"]\n";
34          }
35          return cad;
36      }
37      public String imprimePrimos(){
38          String cad=" ";
39          for (byte j=0;j<=i;j++){
40              if(Metodos.esPrimo((int)datos[j]))cad+=j+" ["+datos[j]+"]\n";
41          }
42          return cad;
43      }
44      public int convertirABinario(){
45
46          int cad = (int)datos[i]%2;
47          datos[i]=(int)datos[i]/2;
48          return cad;
49      }
50      public String imprimeFrecuencia() {
51          String cad = "";
52          byte n=0;
            for (int i=1;i<n;i++) {
54              cad += "*";
55          }
56          return cad;
57      }
58      public String imprimeTriangulo(){
59          String cad = "";
60          for (byte j = 0; j <= i; j++) {
61              cad += j + " [" + datos[j] + "]: "
62                      + Metodos.filasDigitos((int) datos[j]) + "\n";
63          }
64          return cad;
65      }
66      public String imprimesumaDigitos(){
67          String cad=" ";
68          for (byte j = 0; j <= i; j++){
69              Metodos.sumaDigitos((int)datos[j]);
70              cad+=j+" ["+datos[j]+"]\n";
71          }
72           return cad;
73      }
74  }
```

Estos métodos permiten realizar diferentes operaciones sobre el arreglo de datos, como insertar nuevos datos, imprimir los datos almacenados de diferentes formas y realizar cálculos específicos sobre los datos almacenados.

```
1   package TestPrueba;
2
3   import java.util.Random;
4   import EntradaSalida.Tools;
5   import ArrayUnidimen.AlmacenarDatos;
6
7   public class TestAlmacenarDatos {
8       public static void main(String [] args){
9
10          menuArray("Aleatorio,Leer,Constantes,Primos,Triangulo,Frecuencia,"
11                  + "sumaDigitos,Imprimir");
12
13      }
14      public static void menuArray(String menu){
15          AlmacenarDatos dat=new AlmacenarDatos((byte)10);
16
17          int dato;
            String sel= "",cad=" ";
19
```

```
            do{
                sel=Tools.boton(menu);
                switch(sel){
                    case "Aleatorio":
                        Random aleatorio=new Random();
                        dato=aleatorio.nextInt(30);
                        dat.insetardatoLectura(dato);
                        Tools.imprimePantalla("Datos almacenados:\n"+dat.imprimeDatos
                    break;
                    case "Leer":
                        dato=Tools.leerInt(Integer.MIN_VALUE+">Dar un Entero <"+Inte
                        dat.insetardatoLectura(dato);
                        Tools.imprimePantalla("Datos almacenados:\n"+dat.imprimeDatos
                    break;
                    case "Constantes":
                        int a[]={35,-12,41,82,35,247,-71};
                        Tools.imprimePantalla("Datos almacenados:\n"+dat.imprimeDatos
                    break;
                    case "Primos":
                        if(dat.arrayVacio())Tools.salidaError("Array Vacio..insertar
                        cad=dat.imprimePrimos();
                    break;
                   case "Triangulo":
                        if(dat.arrayVacio())Tools.salidaError("Array Vacio..insertar
                        else cad=dat.imprimeTriangulo();
                        break;
                    case "Frecuencia":
                        if(dat.arrayVacio())Tools.salidaError("Array Vacio..insertar
                        else cad=dat.imprimeFrecuencia();
                        break;
                    case "sumaDigitos":
                        if(dat.arrayVacio())Tools.salidaError("Array Vacio..insertar
                        else cad=dat.imprimesumaDigitos();
                        break;
                    case "Imprimir":
                        if(dat.arrayVacio())Tools.salidaError("Array Vacio..insertar
                        else Tools.imprimePantalla("Datos almacenados:\n"+dat.imprim
                    break;
                        case "Salir": break;

                }
            }while(!sel.equalsIgnoreCase("Salir"));
        }
    }
```

Este código permite interactuar con la clase AlmacenarDatos y probar todas las funcionalidades implementadas en ella.

```java
package ArrayUnidimen;

import EntradaSalida.Tools;

public class ArrayObjetos {

    private Rectangulo datos[];
    private byte i;

    public ArrayObjetos(){}
    public ArrayObjetos(byte tam){
        datos=new Rectangulo[tam];
        i=-1;
    }
    public boolean arrayVacio(){
        return i==-1;
    }
    public boolean espacioArray(){
        return i<datos.length-1;
    }
    public Rectangulo crearObjeto(){
        Rectangulo rec=new Rectangulo();

        rec.setAltura(Tools.leerDouble("Altura del rectangulo"));
        rec.setBase(Tools.leerDouble("Base del rectangulo"));
        return rec;
    }
    public void insetardatoLectura(){
        if(espacioArray())
        {
            i++;
            datos[i]=crearObjeto();
        }else
            Tools.salidaError("Array Lleno");
        }
    public String imprimeDatosArray(){

        String cad=" ";
        for(byte j=0;j<=i;j++){
            cad+=j+"["+datos[j].toString()+"]\n";
        }
        return cad;
    }
```

```
44      public void ordenaBurbuja(){
            Rectangulo aux=new Rectangulo();
46          byte k,l;
47          for(k=0;k<datos.length-1;k++){
48              for(l=(byte) (k+1);l<datos.length;l++){
49                  if(datos[k].getArea()>datos[l].getArea()){
50                      aux=datos[k];
51                      datos[k]=datos[l];
52                      datos[l]=aux;
53                  }
54              }
55          }
56      }
57
58
59  }
```

Esta clase proporciona funcionalidades para crear, almacenar, imprimir y ordenar rectángulos en un arreglo.

```
1   package ArrayUnidimen;
2
3   public class Rectangulo {
4       private double Base;
5       private double Altura;
6       private double Area;
7       private double Perimetro;
8
9       public Rectangulo(){}
10
11      public Rectangulo (double Base,double Altura){
12
            this.Area=CalArea();
            this.Perimetro=CalPerimetro();
15          this.Base=Base;
16          this.Altura=Altura;
17
18      }
19      public double getArea() {
20
21          return Area;
22      }
23
24      public double getPerimetro() {
25
```

```
26          return Perimetro;
27      }
28      public double getBase(){
29          return Base;
30      }
31      public void setBase(double Base) {
32          this.Base=Base;
33
34      }
35      public double getAltura(){
36          return Base;
37      }
38      public void setAltura(double Altura) {
39          this.Altura=Altura;
40
41      }
42
43      @Override
        public String toString() {
45          return "Rectangulo{ " + "Base=" + Base + ", Altura=" + Altura
46                  + ", Area=" + CalArea() + ", Perimetro=" + CalPerimetro() + '}';
47      }
48
49      public double CalArea(){
50
51          return (Altura*Base);
52      }
53      public double CalPerimetro(){
54
55          return (2*Altura+Base);
56      }
57  }
```

La clase Rectangulo proporciona funcionalidades para crear y manipular rectángulos, calculando su área y perímetro.

```java
package TestPrueba;


import EntradaSalida.Tools;
import ArrayUnidimen.ArrayObjetos;

public class TestAlmacenarDatos2 {
    public static void main(String [] args){

        menuArrayObjetos("Leer,Burbuja,Imprimir,Salir");

    }
    public static void menuArrayObjetos(String menu){
        ArrayObjetos objetos=new ArrayObjetos((byte)5);

        int dato;
        String sel= "",cad=" ";

        do{
            sel=Tools.boton(menu);
            switch(sel){

                case "Leer":
                    objetos.insetardatoLectura();
                    Tools.imprimePantalla("Datos Almacenados:\n"
                            +objetos.imprimeDatosArray());
                break;
                case "Burbuja":

                case "Imprimir":
                    Tools.imprimePantalla("Datos almacenados:\n"
                            +objetos.imprimeDatosArray());
                    break;

                case "Salir": break;

            }
        }while(!sel.equalsIgnoreCase("Salir"));
    }
}
```

Este programa te permite crear objetos de tipo Rectangulo, almacenarlos en un arreglo y realizar operaciones como la lectura de datos, ordenamiento y visualización de los rectángulos almacenados.

```java
package ArrayUnidimen;

import EntradaSalida.Tools;
import ArrayUnidimen.IMC;

public class ArrayObjetos2 {

    private IMC datos[];
    private byte i;

    public ArrayObjetos2() {
    }

    public ArrayObjetos2(byte tam) {
        datos = new IMC[tam];
        i = -1;
    }

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

    public boolean espacioArray() {
        return i < datos.length - 1;
    }

    public IMC crearObjeto() {
        IMC per = new IMC();

        per.setNomPer(Tools.leerString("Nombre"));
        per.setEdadPer(Tools.leerByte("Edad"));
        per.setSexPer(Tools.leerChar("Género"));
        per.setAlturaPer(Tools.leerFloat("Altura"));
        per.setPesoPer(Tools.leerFloat("Peso"));

        return per;
    }

    public void insetardatoLectura() {
        if (espacioArray()) {
            i++;
            datos[i] = crearObjeto();
        } else {
            Tools.salidaError("Array Lleno");
        }
    }

```

```
48      public String imprimeDatosArray() {
49          String cad = "";
50          for (byte j = 0; j <= i; j++) {
51              cad += j + "[" + datos[j].toString() + "]\n";
52          }
53          return cad;
54      }
55
56      public void ordenaBurbuja() {
            IMC aux = new IMC();
58          byte k, l;
59          for (k = 0; k < datos.length - 1; k++) {
60              for (l = (byte) (k + 1); l < datos.length; l++) {
61                  if (datos[k].getNomPer().compareTo(datos[l].getNomPer()) > 0) {
62                      aux = datos[k];
63                      datos[k] = datos[l];
64                      datos[l] = aux;
65                  }
66              }
67          }
68      }
69
70      public void mostrarEstadisticas() {
71          int totalMasculinos = 0;
72          int totalFemeninos = 0;
73          int totalBajoPeso = 0;
74          int totalSobrepeso = 0;
75          int totalObesidad = 0;
76          int totalObesidadSevera = 0;
77
78          for (int j = 0; j <= i; j++) {
79              IMC persona = datos[j];
80
81              if (persona.getSexPer() == 'M') {
82                  totalMasculinos++;
83              } else if (persona.getSexPer() == 'F') {
84                  totalFemeninos++;
85              }
86
87              String estadoPeso = persona.getEdoPesoPer();
                if (estadoPeso.equals("Bajo peso")) {
89                  totalBajoPeso++;
90              } else if (estadoPeso.equals("Sobrepeso")) {
91                  totalSobrepeso++;
92              } else if (estadoPeso.equals("Obesidad")) {
93                  totalObesidad++;
```

```
                } else if (estadoPeso.equals("Obesidad severa")) {
                    totalObesidadSevera++;
                }
            }

            Tools.imprimePantalla("Estadísticas:\n" +
                    "\nTotal de personas del género masculino: " + totalMasculinos+
                    "\nTotal de personas del género femenino: " + totalFemeninos+
                    "\nTotal de personas bajo de peso: " + totalBajoPeso+
                    "\nTotal de personas con sobrepeso: " + totalSobrepeso+
                    "\nTotal de personas con obesidad: " + totalObesidad+
                    "\nTotal de personas con obesidad severa: " + totalObesidadSever
        }
        public void consultaIndividual() {
        byte indice = Tools.leerByte("Ingrese el índice de la persona a consultar (0

        if (indice >= 0 && indice <= i) {
            IMC persona = datos[indice];

            Tools.imprimePantalla("\nDatos de la persona en la posición " + indice +
                    "Nombre: " + persona.getNomPer() + "\n" +
                    "Género: " + persona.getSexPer() + "\n" +
                    "Edad: " + persona.getEdadPer() + "\n" +
                    "Altura: " + persona.getAlturaPer() + " cm\n" +
                    "Peso: " + persona.getPesoPer() + " kg\n" +
                    "Estado de peso: " + persona.getEdoPesoPer() + "\n" +
                    "IMC: " + persona.getImcPer());
        } else {
            Tools.imprimePantalla("Índice inválido.");
            }
        }
    }
```

Estos métodos te permiten interactuar con el arreglo de objetos IMC y realizar diversas operaciones relacionadas con el cálculo del índice de masa corporal y el análisis de los datos almacenados.

```java
package ArrayUnidimen;

public class IMC {

		private static int Folio=1000;
		private  String nomPer;
		private byte edadPer;
		private char sexPer;
		private float pesoPer;
		private float alturaPer;
		private String edoPesoPer;
		private float imcPer;

	public IMC(){}

	public IMC(String nomPer, byte edadPer, char sexPer, float pesoPer,
		float alturaPer) {

			this.nomPer = nomPer;
			this.edadPer = edadPer;
			this.sexPer = sexPer;
			this.pesoPer = pesoPer;
			this.alturaPer = alturaPer;
			this.edoPesoPer=edoPeso();
			this.imcPer=calcularIMC();
			Folio++;
		}

	public static int getFolio() {
		return Folio;
	}

	public String getNomPer() {
		return nomPer;
	}

	public byte getEdadPer() {
		return edadPer;
	}

	public char getSexPer() {
		return sexPer;
	}


	public float getPesoPer() {
		return pesoPer;
	}

```

```
50          public float getAlturaPer() {
51                  return alturaPer;
52          }
53
54          public String getEdoPesoPer() {
55                  return edoPesoPer;
56          }
57
58          public float getImcPer() {
59                  return imcPer;
60          }
61
62
63          public void setNomPer(String nomPer) {
64                  this.nomPer = nomPer;
65          }
66
67          public void setEdadPer(byte edadPer) {
68                  this.edadPer = edadPer;
69          }
70
71          public void setSexPer(char sexPer) {
72                  this.sexPer = sexPer;
73          }
74
75          public void setPesoPer(float pesoPer) {
76                  this.pesoPer = pesoPer;
77          }
78
79
80          public void setAlturaPer(float alturaPer) {
81                  this.alturaPer = alturaPer;
82          }
83
84
85          public float calcularIMC () {
86                  return  (pesoPer/ (alturaPer/100));
87          }
88
89      @Override
        public String toString() {
91          return "IMC{" + "nomPer=" + nomPer + ", edadPer=" + edadPer
92                  + ", sexPer=" + sexPer + ", pesoPer=" + pesoPer
93                  + ", alturaPer=" + alturaPer + ", edoPesoPer=" + edoPesoPer
94                  + ", imcPer=" + imcPer + '}';
95      }
96
97    public String edoPeso() {
```

```
98
99                  if(imcPer<18.5) return("Bajo peso");
100                 else if (imcPer>=18.5&& imcPer<=24.9) return("peso normal");
101
102                 else if (imcPer>=25.0 && imcPer<=29.9) return("Sobrepeso");
103
104                 else if (imcPer>=30.0 && imcPer<=34.9)  return("Obesidad");
105                 else if (imcPer>=35.0 && imcPer<=39.0)  return("Obesidad severa"
106             return ("Imc fuera de rango");
107     }
108
109 }
```

Esta clase te permite crear objetos IMC para representar personas y realizar cálculos relacionados con el IMC y el estado de peso.

```java
package TestPrueba;


import EntradaSalida.Tools;
import ArrayUnidimen.ArrayObjetos2;

public class TestAlmacenarDatos3 {
    public static void main(String [] args){

        menuArrayObjetos("Leer,Burbuja,Imprimir,Estadísticas,Consulta,Salir");

    }
    public static void menuArrayObjetos(String menu){
        ArrayObjetos2 objetos=new ArrayObjetos2((byte)5);

        int dato;
        String sel= "",cad=" ";

        do{
            sel=Tools.boton(menu);
            switch(sel){

                case "Leer":
                    objetos.insetardatoLectura();
                    break;

                case "Burbuja":
                    objetos.ordenaBurbuja();
                    Tools.imprimePantalla("Datos ordenados por nombre."
                            +objetos.imprimeDatosArray());
                    break;

                case "Imprimir":
                    Tools.imprimePantalla("Datos almacenados:\n"
                            + objetos.imprimeDatosArray());
                    break;

                case "Estadísticas":
                    objetos.mostrarEstadisticas();
                    break;

                case "Consulta":
                    objetos.consultaIndividual();
                    break;

                case "Salir":
                    break;
            }
        } while (!sel.equalsIgnoreCase("Salir"));
    }
```

Este código proporciona un menú interactivo que te permite realizar diversas acciones, como ingresar datos, ordenarlos, mostrar estadísticas y realizar consultas individuales sobre los datos almacenados en los objetos IMC.

CONCLUSIONES

En este informe, hemos explorado los métodos relacionados con los arreglos unidimensionales y multidimensionales en programación. Estos temas son fundamentales para el manejo eficiente y estructurado de datos en diferentes contextos.

En cuanto a los arreglos unidimensionales, hemos aprendido a crear, acceder y manipular elementos en un solo eje. Además, hemos explorado algoritmos de búsqueda y ordenamiento que nos permiten realizar operaciones específicas en estos arreglos. Los arreglos unidimensionales son útiles cuando necesitamos almacenar y trabajar con conjuntos de datos lineales.

Por otro lado, los arreglos multidimensionales nos han permitido trabajar con datos organizados en múltiples ejes, como matrices o tablas. Hemos aprendido a crear, acceder y manipular elementos en diferentes dimensiones. Los arreglos multidimensionales son especialmente útiles cuando necesitamos representar y procesar datos estructurados en forma de filas y columnas.

En resumen, los métodos relacionados con los arreglos unidimensionales y multidimensionales son herramientas esenciales en la programación. Nos brindan la capacidad de almacenar, acceder y manipular datos de manera eficiente. Comprender estos métodos nos ayuda a desarrollar aplicaciones más robustas y efectivas, permitiéndonos trabajar con conjuntos de datos de diferentes dimensiones y optimizando el rendimiento de nuestros programas.

En futuros proyectos, es importante considerar la selección adecuada de estructuras de datos, incluyendo arreglos unidimensionales y multidimensionales, para garantizar un diseño eficiente y una manipulación adecuada de los datos.

Además, seguir explorando algoritmos y técnicas avanzadas nos permitirá mejorar nuestras habilidades en el manejo de arreglos y su aplicación en diferentes problemas de programación.

BIBLIOGRAFÍA

Stroustrup, B. (2013). C++ Programming Language: The Definitive Reference (C++ Reference Series). Addison-Wesley Professional.

Schildt, H. (2017). Java: The Complete Reference. McGraw-Hill Education.

Matthes, E. (2019). Python Crash Course: A Hands-On, Project-Based Introduction to Programming. No Starch Press.

Wiener, R., & Pinson, L. J. (2017). Fundamentals of OOP and Data Structures in Java. Springer.

Mueller, J. P. (2017). C# 7.0 All-in-One For Dummies. For Dummies.