TECNOLÓGICO NACIONAL DE MÉXICO

INSTITUTO TECNOLÓGICO NACIONAL DE MEXICO CAMPUS ORIZABA

Materia:

Estructura De Datos

Maestra

Martínez Castillo María Jacinta

Integrantes:

Bravo Rosas Yamileth-21010175
Crescencio Rico José Armando - 21010001

Grupo:

3PM – 4PM HRS

Clave:

3a3A

Especialidad:

Ing. informática

Fecha De Entrega

17/04/2023

REPORTE DE PRACTICAS

UNIDAD 2

Introducción.

El presente reporte habla sobre como la recursividad es una técnica fundamental en programación que permite a una función llamarse a sí misma para resolver un problema de manera eficiente y elegante. Se basa en la idea de que un problema se puede descomponer en subproblemas más pequeños y similares al problema original, lo que a su vez permite que la solución se obtenga de forma iterativa. Los procedimientos recursivos son el resultado de aplicar esta técnica en la resolución de problemas.

En este reporte de prácticas, se explorará en detalle la recursividad y los procedimientos recursivos, examinando ejemplos prácticos y discutiendo las ventajas y desventajas de su uso. Se presentarán diferentes estrategias y técnicas para trabajar con la recursividad, con el objetivo de ayudar a desarrollar habilidades prácticas en esta área clave de la programación.

Además, se discutirán las implicaciones de la recursividad en la complejidad temporal y espacial de los algoritmos, y se presentarán herramientas para analizar la complejidad de los procedimientos recursivos. Con este informe, el se tendrá una comprensión sólida de la recursividad y estará equipado para aplicarla de manera efectiva en la resolución de problemas en la programación.

Competencia especifica.

Especifica(s):

Aplica la recursividad en la solución de problemas valorando su pertinencia en el uso eficaz de los recursos.

Genéricas:

- Habilidad para buscar y analizar información proveniente de fuentes diversas.
- Capacidad de análisis y síntesis.
- Habilidad en el manejo de equipo de cómputo.
- Capacidad para trabajar en equipo.
- Capacidad de aplicar los conocimientos en la práctica.

Marco teórico.

Definición:

La recursividad se aplica en una función en la que queremos resolver un problema determinado a la que se denomina función recursiva, esta función solo sabe resolver un caso base, si no recibe en sus parámetros ese caso base entonces descompone el problema en uno mas pequeño hasta llegar a ese caso base que la función puede resolver.

La recursividad en ocasiones puede parecer un proceso infinito o interminable, esto es porque no se entiende por completo como funciona la recursividad, en ocasiones muchos de los problemas de los recién iniciados en la recursividad es olvidar poner un caso base.

Procedimientos iterativos

Las instrucciones de repetición, de iteración o bucles, facilitan la repetición de un bloque de instrucciones, un número determinado de veces o mientras se cumpla una condición.

Por lo general, existen dos tipos de estructuras iterativas o bucles en los lenguajes de programación. Encontraremos un tipo de bucle que se ejecuta un número preestablecido de veces, que es controlado por un contador o índice, incrementado en cada iteración. Este tipo de bucle forma parte de la familia for.

Por otro lado, encontraremos un tipo de bucle que se ejecuta mientras se cumple una condición. Esta condición se comprueba al principio o el final de la construcción. Esta variante pertenece a la familia while or repeat, respectivamente.

Procedimientos recursivos

Un procedimiento o función recursiva es aquella que se llama así misma. Esta permite una repetición recursiva sin un procedimiento diferente de parámetros. La recursión es una alternativa a la iteración muy elegante en la solución de problemas, especialmente si estos tienen naturaleza recursiva. Normalmente, una solución recursiva es menos eficiente en términos de tiempo de computadora, que una solución iterativa debido al tiempo adicional de llamada a procedimientos. En

muchos casos, la recursión especificar permite una solución más simple y natural para resolver un problema que en otro será difícil. Por esta razón recursión (recursividad) es una herramienta muy potente para la resolución de problemas de programación.

Tipos de recursividad

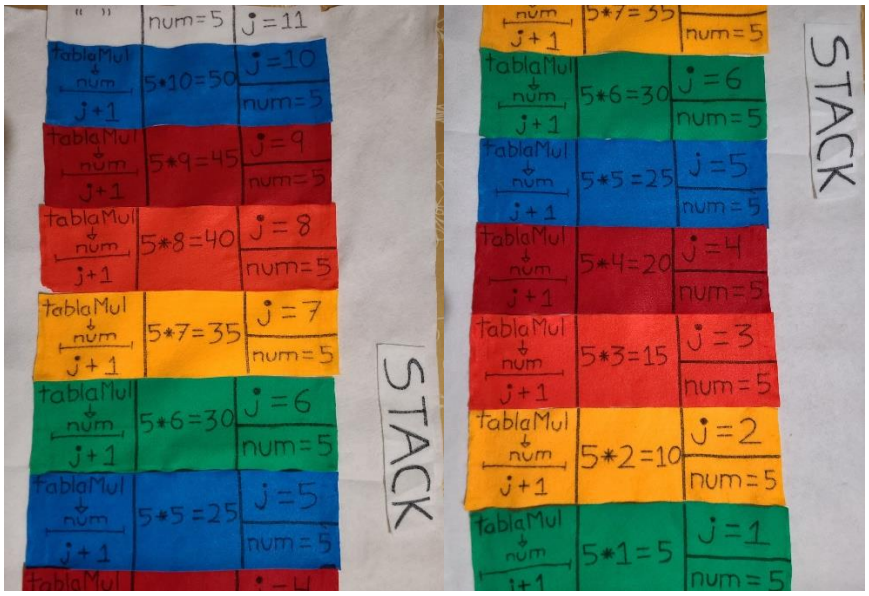
Según el subprograma al que, se llama, existen dos tipos de recursión. -Recursividad simple o directa: La función incluye una referencia a si misma.

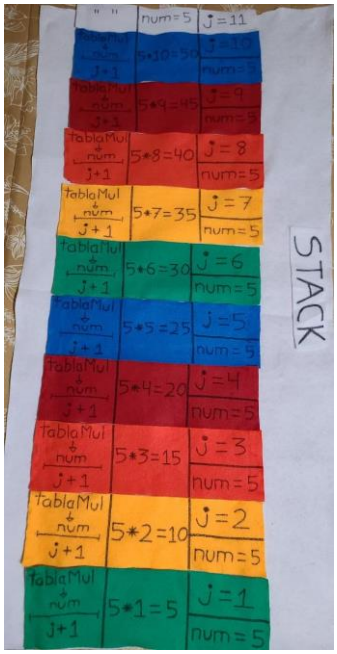
-Recursividad o indirecta: El módulo llama a otros módulos de forma ya la última llamada se llama al primero.

Iteración	Recursividad
Se realiza mediante el uso de bucles o ciclos	Se realiza mediante la llamada a una función dentro de sí misma
Es una solución más directa y simple para problemas sencillos	Es una solución más compleja para problemas complejos
Puede ser más eficiente que la recursividad en términos de velocidad de ejecución	Puede ser menos eficiente que la iteración en términos de velocidad de ejecución
Es más fácil de entender y depurar	Puede ser más difícil de entender y depurar
Requiere menos memoria para ejecutar	Puede requerir más memoria para ejecutar
El número de iteraciones necesarias puede ser fácilmente controlado y ajustado	El número de veces que se llama a una función recursiva puede ser más difícil de controlar y ajustar
Es adecuado para problemas que se pueden resolver de forma lineal, es decir, que sigue una secuencia lógica predecible	Es adecuado para problemas que se pueden dividir en subproblemas más pequeños y resolver de forma recursiva
Puede ser utilizado en cualquier lenguaje de programación	Algunos lenguajes de programación pueden tener limitaciones en cuanto a la recursividad y su uso puede requerir cuidado especial

Hicimos un trabajo con diferentes materiales donde explicamos un solo programa de forma recursiva nosotros presentamos el de tabla de multiplicar y aquí presentamos evidencia de nuestro trabajo.

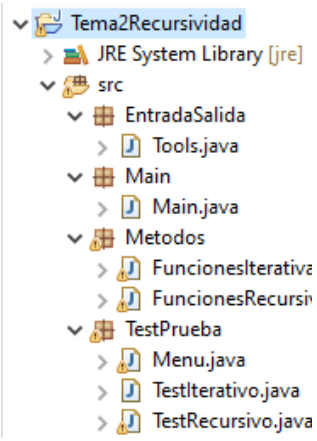
```
public static String tablaMul(int num, int j)
{
    if(j<=10)
    {
        return +num+"*"+"="+(num*j)+"\n"+tablaMul(num,(j+1));
    }
    else
        return "";
}
```





Desarrollo de la practica:

Creamos un nuevo proyecto sobre Tema2 de Recursividad donde se encuentran cuatro paquetes



El primero se trata de la entrada y salidas donde se alojan la clase de los tools estas son las herramientas informáticas , son programas, instrucciones usadas para efectuar otras tareas de modo más sencillo.

```
package EntradaSalida;

import javax.swing.JOptionPane;

public class Tools {

    public static byte leerByte(String msje){
          return(Byte.parseByte(JOptionPane.showInputDialog(null,msje,"Lectura Byte",JOptionPane.QUESTION_MESSAGE)));
       }

       public static int leerInt(String msje){
           return(Integer.parseInt(JOptionPane.showInputDialog(null,msje,"Lectura Int",JOptionPane.QUESTION_MESSAGE)));
       }

       public static String leerString(String msje){
           return(JOptionPane.showInputDialog(null,msje,"Lectura String",JOptionPane.QUESTION_MESSAGE));
       }

       public static float leerFloat(String msje){
           return(Float.parseFloat(JOptionPane.showInputDialog(null,msje,"Lectura Float",JOptionPane.QUESTION_MESSAGE)));
       }

       public static char leerChar(String msje){
           return(JOptionPane.showInputDialog(null,msje,"Lectura Char",JOptionPane.QUESTION_MESSAGE).charAt(0));
       }

       public static short leerShort(String msje){
           return(Short.parseShort(JOptionPane.showInputDialog(null,msje,"Lectura Short",JOptionPane.QUESTION_MESSAGE)));
       }

       public static double leerDouble(String msje){
           return(Double.parseDouble(JOptionPane.showInputDialog(null,msje,"Lectura Double",JOptionPane.QUESTION_MESSAGE)));
       }

       public static long leerLong(String msje){
           return(Long.parseLong(JOptionPane.showInputDialog(null,msje,"Lectura Long",JOptionPane.QUESTION_MESSAGE)));
       }
```

```
        public static void imprimePantalla(String msje){
            JOptionPane.showMessageDialog(null,msje,"Salida",JOptionPane.INFORMATION_MESSAGE);
        }
        public static void imprimeError(String msje){
            JOptionPane.showMessageDialog(null,msje,"Error",JOptionPane.INFORMATION_MESSAGE);


        }
        public static String menuDesplegable() {
            String valores[] = {"MenuIterativo","MenuRecursivo","Salir"};
            String res = (String)JOptionPane.showInputDialog(null,"M E N U","Selecciona opcion:",JOptionPane.QUESTION_MESSAGE,null,valores,valores[0
            return(res);
        }
        public static String desplegableIterativo() {
            String valores[] = {"usoDoWhile","usoFor","usoWhile","tablaDeMultiplicar","printArray","sumaDiviciones","decimalOctal","potencias","bina
            String res = (String)JOptionPane.showInputDialog(null,"M E N U","Metodos Iterativos:",JOptionPane.QUESTION_MESSAGE,null,valores,valores[
            return(res);
        }
        public static String desplegableRecursivo() {
            String valores[] = {"funcionIterativa","funcionIterativa2","funcionIterativa3","tablaDeMultiplicar","printArray","sumaDiviciones","decim
            String res = (String)JOptionPane.showInputDialog(null,"M E N U","Metodos Recursivos:",JOptionPane.QUESTION_MESSAGE,null,valores,valores[
            return(res);
        }

public static void ImprimeMsje(String msje){
    JOptionPane.showMessageDialog(null,msje,"Salida",JOptionPane.INFORMATION_MESSAGE);
}
public static void ImprimeErrorMsje(String msje){
    JOptionPane.showMessageDialog(null,msje,"Salida",JOptionPane.INFORMATION_MESSAGE);
}
public static String desplegable(String menu) {
    String valores[]=menu.split(",");
    String res=(String)JOptionPane.showInputDialog(null,"M E N U"," Selecciona opcion:",JOptionPane.QUESTION_MESSAGE,null,valores,valores[0]);
    return(res);
    }
}
```

El segundo paquete contiene una clase main que esta solo manda a llamar a la clase menú.

```
1  package Main;
2  import TestPrueba.Menu;
3  public class Main {
4
5  public static void main(String[] args) {
6
7          Menu.menu();
8
9      }
10 }
11
```

El tercero es el paquete métodos Esta contiene dos clases una de funciones iterativas y la otra clase fue de funciones recursivas estas contienen todos los métodos iterativos y estos fueron convertidos a recursivos

Esta clases es función iterativa

```java
package Metodos;
import javax.swing.JOptionPane;
public class FuncionesIterativas {

    public static void usoFor(int n)
    { //Valor inicial, Condicion true; Incremento
        for (int j =1; j<=n;j++)
        {
            Tools.imprimePantalla("" + j);
        }
    }

    public static void usoDoWhile(int n)
    {
        String cad = "";
        int j = 1; //Valor Inicial
        do //Condicion True
        {
            cad+="\nIncremento: " + j;
            j++; //Incremento
        }while(j<=n);
        Tools.imprimePantalla(cad);
    }

    public static void usoWhile(int n)
    {
        String cad ="";
        int j = 1; //Valor Inicial
        while(j<=n) //Condicion True
        {
            cad+="\nIncremento: " + j;
            j++; //Incremento
        }
       Tools.imprimePantalla(cad);
    }

```

```java
public static void tablaMultiplicar(int num)
{
    String tabla="";
    for (int j=1; j<=10; j++) {
        tabla = num+"*"+j+"="+(num*j);
        Tools.imprimePantalla(tabla);

    }
}

public static void imprimeArray(int a [])
{
    String lista="";
    for(byte j=0;j<a.length;j++){

        lista+=j+"["+a[j]+"]\n";
    Tools.imprimePantalla("\n"+lista);
}
}

public static int SumaDiviciones(int dato)
{
    int k=1,suma=0;
    do {
        if(dato % k==0)
            suma+=k;
        k++;

    }
    while (k<dato);

    return suma;
```

```java
public static int decimalAOctal (int decimal)
    {
        int Octal = 0, i = 1;
        while (decimal!=0)
        {
            Octal= Octal + (decimal % 8)*i;
            decimal=decimal /8;
            i = i * 10;
        }
        return Octal;
}

public static int binarioAdecimal(int binario) {
    int decimal = 0;
    int potencia = 0;
    while (binario!=0){
        decimal += (binario % 10) * Math.pow(2, potencia);
        binario = binario / 10;
        potencia++;
    }
    return decimal;
}

public static String fibonacci(int contador)
{
    int Fib1 =0;
    int Fib2 =1;
    String cad="";
    int numero = 0;
    for (int i = 1; i <= contador; i++) {
        cad+= "\n"+numero;
        numero = Fib1 + Fib2;
        Fib1 = Fib2;
        Fib2 = numero;
    }
    return cad;
}
```

```java
public static void ordenaBurbuja(int datos[])
{
    int i ,j, aux;

    for (i=0;i<datos.length-1;i++)
        for(j=i+1;j<datos.length;j++) {
            if (datos[i]<datos[j]) {//canbia el orden mayor o menor
                aux=datos[i];
                datos[i]=datos[j];
                datos[j]=aux;
            }
        }
  }

public static String verArray (int datos[])
{
    String cad="";

    for (int k=0;k<datos.length;k++) {
        cad+=k+"["+datos[k]+"]"+"\n";
    }
    return cad;
}

public static double factorial(int dato)
{
    int f=1,c=1;
    while (c<=dato)
    {
        f*=c;
        c++;
    }

    return f;

}
```

```java

    public static void leerMatriz(int a[][])
    {
        int i,j;
        String ca="";
        for( i=0; i<a.length;i++)
        for( j=0; j<a[0].length;j++)
        {
         a[i] [j]=Integer.parseInt(JOptionPane.showInputDialog("Dato:"));
        }


}
```

Esta clase es de funciones recursivas

```java
package Metodos;
import javax.swing.JOptionPane;
import EntradaSalida.Tools;
public class FuncionesRecursivas {

    public static String funcionIterativa(int n)
    {
        return (n > 10)
                ? ""
                : "incremento" + n + "\n" + funcionIterativa(n + 1);

    }

    public static String funcionIterativa2(int j, int n)
    {
        return (j <= n)
                ? "\nIncremento: " + j + funcionIterativa2(j + 1, n)
                : "";
    }

    public static String funcionIterativa3(int j, int n)
    {
        return(j<=n)? "\nIncremento"+j+funcionIterativa3(j+1,n):"";
    }


    public static String tablaMul(int num, int j)
    {
        if(j<=10)
        {
            return +num+"*"+"="+(num*j)+"\n"+tablaMul(num,(j+1));
        }
        else
            return "";
    }
```

```java
public static String ArrayRecursivo(int[] array, int index) {
    return (index >= array.length)
        ? ""
        : index + "[" + array[index] + "]\n"  + ArrayRecursivo(array, index + 1);
}


public static int SumaDivi(int dato,int k)
{
    int suma=0;
    if(k<dato){
        if(dato % k==0)
            suma+=k;
        return suma+SumaDivi(dato,(int)(k+1));
    }
    return suma;
}

public static int deciOctal(int nOctal,int cont, int decimal)
{
    if(decimal!=0)
        return deciOctal((nOctal + ((decimal % 8) * cont)), cont * 10, decimal / 8);
    return nOctal;
}


public static int potencias(int digit, int pow) {
    return (pow == 0)
        ? 1
        : digit * potencias(digit, pow - 1);
}


  public static String fibonacci(int contador, int nfib1, int nfib2, int numero, int i)
    {
        if(i<=contador)
            return "\n"+ numero+fibonacci(contador, nfib2, (nfib2+nfib1), (nfib1+nfib2), i+1);
        else return "";
    }
```

```java
//1 al 20 valor en factorial
public static String ValorFact(int numero, long factorial) {
    return (numero <=  20)
        ? numero + " = " + factorial + "\n" + ValorFact(numero + 1, factorial * (numero + 1))
        : "";
}


public static double factorialRecur(int dato , int c)
{
    if (dato ==0|| dato==1 )return 1;

else
    if(c<=dato)
    return c*factorialRecur(dato, c+1);
    else return 1;
}

public static void LeerMatriz(int[][] array, int i, int j) {
    if (i < array.length) {
        if (j < array[0].length) {
            array[i][j] = Integer.parseInt(JOptionPane.showInputDialog("dato: "));
            LeerMatriz(array, i, j + 1);
        } else {
            LeerMatriz(array, i + 1, 0);
        }
    }
}
```

El cuarto paquete es el de test prueba este contiene tres clases dos de ellas contiene el menú de funciones iterativas y la otra de recursivas contiene una lista de opciones que los usuarios pueden seleccionar para realizar una tarea o acceder a una función específica del programa. El menú está ubicado en la de menú principal donde manda a llamar a la clase test iterativo y test recursivos del programa es un menú desplegable.

Clase test Iterativo contiene el menú de la función iterativa

```java
package TestPrueba;
import EntradaSalida.Tools;
import Metodos.FuncionesIterativas;
public class TestIterativo {
    public static void menu() {
        String sel = "usoDoWhile,usoFor,usoWhile,tablaDeMultiplicar,imprimeArray,sumaDiviciones,"
                + "decimalOctal,potencias,binarioDecimal,fibonacci,ordenaBurbuja,showArray,factorial";

        do {
            sel = Tools.desplegable(sel);
            switch (sel) {
            case "usoDoWhile":
                FuncionesIterativas.usoDoWhile(Tools.LeerInt("Indica un limite"));
                break;
            case "usoFor":
                FuncionesIterativas.usoFor(Tools.LeerInt("Indica un limite"));
                break;
            case "usoWhile":
                FuncionesIterativas.usoWhile(Tools.LeerInt("Indica un limite"));
                break;
            case "tablaDeMultiplicar":
                FuncionesIterativas.tablaMultiplicar(Tools.LeerInt("Tabla de multiplicar"));
                break;
            case "imprimeArray":
                int array[] = {1,2,3,4,5,6,7,8,9,0};
                FuncionesIterativas.imprimeArray(array);;
                break;
            case "sumaDiviciones":
                int sumaDiviciones = FuncionesIterativas.SumaDiviciones(Tools.LeerInt("Ingresa dato"));
                Tools.imprimePantalla("Respuesta: " + sumaDiviciones);
                break;
            case "decimalOctal":
                int decimal = Tools.LeerInt("Dato a convertir:");
                int octal = FuncionesIterativas.decimalAOctal(decimal);
                Tools.imprimePantalla(decimal + " a octal es: " + octal);
                break;

            case "binarioDecimal":
                int binario = Tools.LeerInt("Ingresa binario");
                int binarioDecimal = FuncionesIterativas.binarioADecimal(binario);
                Tools.imprimePantalla(binario + " a decimal: " + binarioDecimal);
                break;
            case "fibonacci":
                int numeroEjecucion = Tools.LeerInt("Ingresa el numero de veces");
                Tools.imprimePantalla(FuncionesIterativas.fibonacci(numeroEjecucion));
                break;
            case "ordenaBurbuja":
                int burbja[] = {12,23,54,56,76,1,2,3,100};
                Tools.imprimePantalla("Array desordenado: " + FuncionesIterativas.verArray(burbja));
                FuncionesIterativas.ordenaBurbuja(burbja);
                Tools.imprimePantalla("Array ordenado: " + FuncionesIterativas.verArray(burbja));
                break;
            case "VerArray":
                int burbja2[] = {12,23,54,56,76,1,2,3,100};
                 Tools.imprimePantalla(FuncionesIterativas.verArray(burbja2));
                break;
            case "factorial":
                int dato = Tools.LeerInt("Ingresa un dato");
                Tools.imprimePantalla("Factorial de" + dato + " = " + FuncionesIterativas.factorial(dato));
                break;
            case "leerMatriz":
                break;
            }
        } while (!sel.equalsIgnoreCase("Salir"));
    }
```

Clase test Recursivo contiene el menú de las funciones recursivas

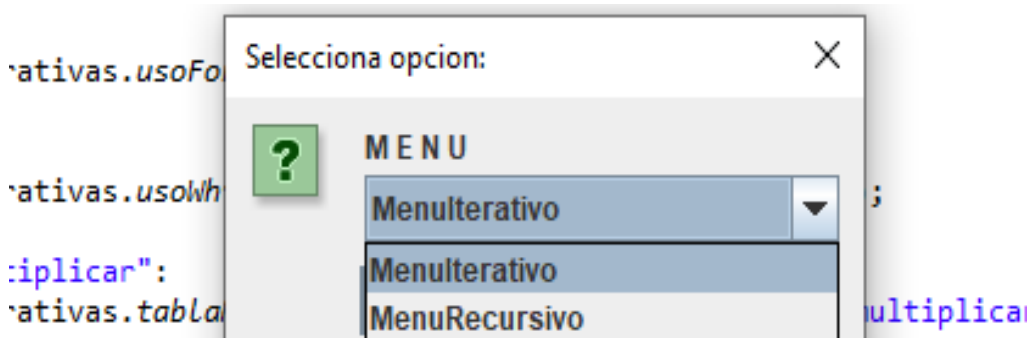
```java
package TestPrueba;
import EntradaSalida.Tools;
public class TestRecursivo {
            public static void menu() {
                String sel = "funcionIterativa,funcionIterativa2,funcionIterativa3,tablaDeMultiplicar,imprimeArray,"
                        + "sumaDiviciones,decimalOctal,potencias,fibonacci,ordenaBurbuja,showArray,factorial";
                int i=0, j=0, digit,num=0;
                do {
                    //sel = Tools.desplegable(sel);
                    sel = Tools.desplegableRecursivo();
                    switch (sel) {
                    case "funcionIterativa":
                        i = Tools.leerInt("Inicio del contador, ten en centa que se ejecuta hasta el 10");
                        Tools.imprimePantalla(FuncionesRecursivas.funcionIterativa(i));
                        break;
                    case "funcionIterativa2":
                        i = Tools.leerInt("Inicio del contador: ");
                        j = Tools.leerInt("Ingresa limite del contador");
                        Tools.imprimePantalla(FuncionesRecursivas.funcionIterativa2(i, j));
                        break;
                    case "funcionIterativa3":
                        i = Tools.leerInt("Inicio del contador: ");
                        j = Tools.leerInt("Ingresa limite del contador");
                        Tools.imprimePantalla(FuncionesRecursivas.funcionIterativa2(i, j));
                        break;
                    case "tablaDeMultiplicar":
                        num= Tools.leerInt("Digito de la tabla: ");
                        Tools.imprimePantalla(FuncionesRecursivas.tablaMul(num, j));
                        break;
                    case "imprimetArray":
                        int array[] = {1,2,3,4,5,6,7,8,9,0};
                        Tools.imprimePantalla(FuncionesRecursivas.ArrayRecursivo(array, 0));
                        break;
                    case "sumaDiviciones":
                        digit = Tools.leerInt("Ingresa dato");
                        int sumaDiviciones = FuncionesRecursivas.SumaDivi(digit, 1);
                        Tools.imprimePantalla("La suma de los divisores de " + digit + " es " + sumaDiviciones);
                        break;
                    case "decimalOctal":
                        int decimal= Tools.leerInt("Dato a convertir:");
                        String octal = FuncionesRecursivas.decimalOctal(decimal);
                        Tools.imprimePantalla(decimal + " a octal es: " + octal);
                        break;

                    case "potencias":
                        digit = Tools.leerInt("Digito: ");
                        int pow = Tools.leerInt("potencia: ");
                        int result = FuncionesRecursivas.potencias(digit, pow);
                        Tools.imprimePantalla(digit + " potencia de " + pow + " = " + result);
                        break;

                    case "fibonacci":
                        int ejecuciones = Tools.leerInt("Veces en las que se ejecuta:");
                        int f1 = Tools.leerInt("Numero fibonacci");
                        int f2 = Tools.leerInt("Numero fibonacci2");
                         Tools.imprimePantalla(FuncionesRecursivas.fibonacci(ejecuciones, f1, f2, 1, 1));
                        break;

                    case "showArray":
                        int burbja2[] = {12,23,54,56,76,1,2,3,100};
                        Tools.imprimePantalla(FuncionesRecursivas.ValorFact(num, i));
                        break;
                    case "factorial":
                        int dato = Tools.leerInt("Ingresa un dato");
                        Tools.imprimePantalla("Factorial de" + dato + " = " + FuncionesRecursivas.factorialRecur(dato, 1));
                        break;
                    }
                } while (!sel.equalsIgnoreCase("Salir"));
```

Clase menú manda a llamar a las clases test para poder desplegar el menú

```
package TestPrueba;

import EntradaSalida.Tools;

public class Menu {
public static void menu() {

        String sel = "";

        do {
            sel = Tools.menuDesplegable();
            switch (sel) {
            case "MenuIterativo":
                TestIterativo.menu();
                break;
            case "MenuRecursivo":
                TestRecursivo.menu();
                break;
            default:
                Tools.imprimePantalla("Saliendo...");
            }
        } while (!sel.equalsIgnoreCase("Salir"));
    }
}
```

Resultados:

Creamos una serie de programas iterativo y los convertimos a recursivos igual creamos un menú don de incluimos a los dos tipos

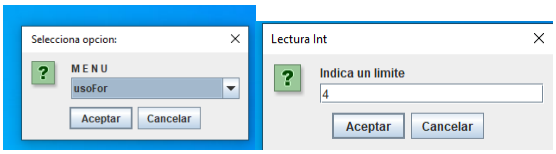


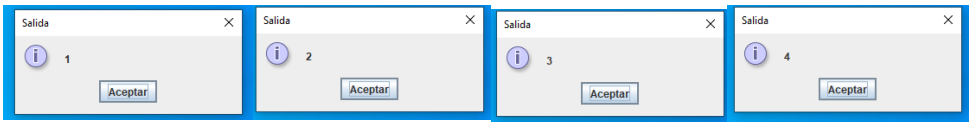
Menú Iterativas:

permiten repetir una sentencias o conjunto de ellas.

Uso de for: Se ejecuta un número preestablecido de veces, que es controlado por un contador, incrementado en cada iteración.

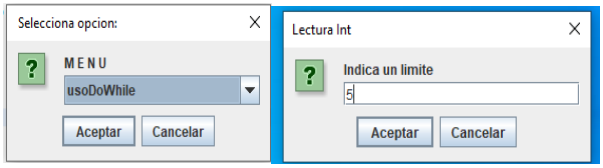
```java
public static void usoFor(int n)
{ //Valor inicial, Condicion true; Incremento
    for (int j =1; j<=n;j++)
    {
        Tools.imprimePantalla("" + j);
    }
}
```

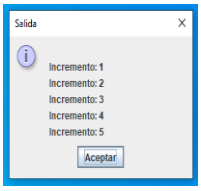




Uso de DoWhile: Este método utiliza la estructura de control de flujo "do-while" para iterar desde un valor inicial de 1 hasta "n", incrementando en 1 en cada iteración y concatenando una cadena de texto que representa el valor de "j" en cada iteración.

```java
public static void usoDoWhile(int n)
{
    String cad = "";
    int j = 1; //Valor Inicial
    do //Condicion True
    {
        cad+="\nIncremento: " + j;
        j++; //Incremento
    }while(j<=n);
    Tools.imprimePantalla(cad);
}
```





Uso de While: Este método utiliza la estructura de control "while" para iterar desde un valor inicial de 1 hasta "n" e imprimir una cadena de texto que representa el valor actual de "j" en cada iteración.

```java
public static void usoWhile(int n)
{
    String cad ="";
    int j = 1; //Valor Inicial
    while(j<=n) //Condicion True
    {
        cad+="\nIncremento: " + j;
        j++; //Incremento
    }
   Tools.imprimePantalla(cad);
}
```

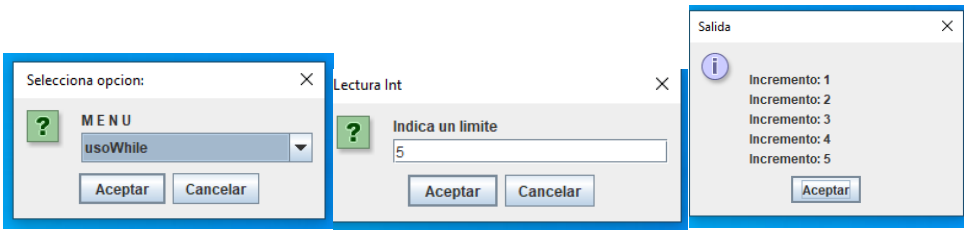
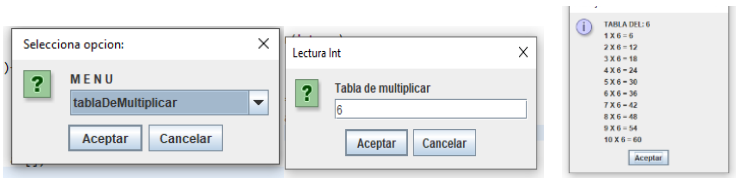


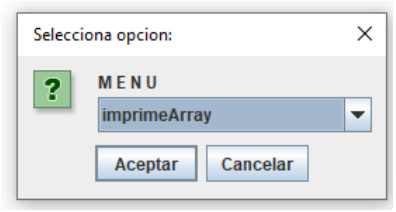
Tabla de multiplicar: utiliza un enfoque iterativo mediante el uso de bucles "for" para imprimir la tabla de multiplicar completa de un número dado desde 1 hasta 10.

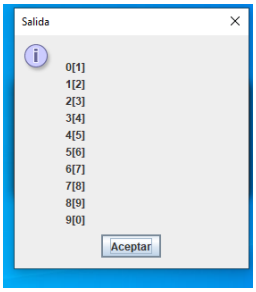
```
public static void tablaMultiplicar(byte num)
{
    String tabla="";
    for (int j=1; j<10; j++) {
        tabla= num+"*"+j+"="+(num*j)+"\n";
    }
    Tools.imprimePantalla(tabla);
}
```



Imprime Array: toma una matriz de enteros como entrada e imprime cada elemento de la matriz junto con su índice.

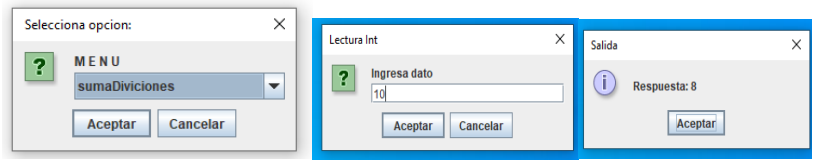
```
public static void imprimeArray(int a [])
{
    String lista="";
    for(byte j=0;j<a.length;j++){

        lista+=j+"["+a[j]+"]\n";
    Tools.imprimePantalla("\n"+lista);
}
}
```





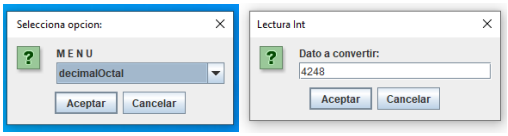
Suma de divisiones: toma una entrada de número entero "dato" y devuelve una salida de número entero "suma". El método calcula la suma de todos los divisores del entero de entrada "dato".

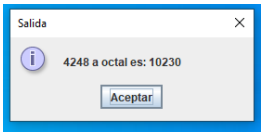
```
public static int SumaDiviciones(int dato)
{
    int k=1,suma=0;
    do {
        if(dato % k==0)
            suma+=k;
        k++;

    }
    while (k<dato);

    return suma;
}
```



Decimal a Octal: toma un valor entero en formato decimal como entrada y lo convierte en un formato octal equivalente.

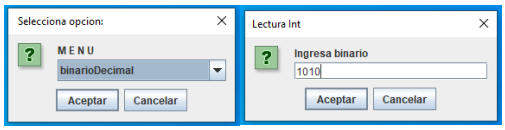
```java
public static int decimalAOctal (int decimal)
    {
        int Octal = 0, i = 1;
        while (decimal!=0)
        {
            Octal= Octal + (decimal % 8)*i;
            decimal=decimal /8;
            i = i * 10;
        }
        return Octal;
}
```

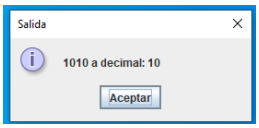




Binario a Decimal: Este es un método Java que convierte un número binario representado como un entero a su equivalente decimal. El método toma un valor entero binario como entrada, que se supone que representa un número binario, y devuelve un valor entero que representa el equivalente decimal del número binario.

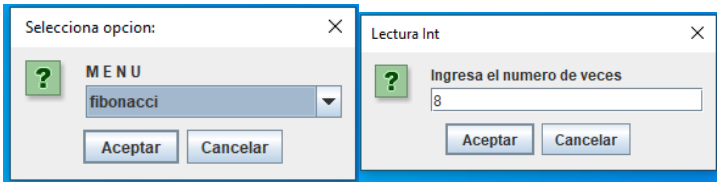
```java
}
public static int binarioAdecimal(int binario) {
    int decimal = 0;
    int potencia = 0;
    while (binario!=0){
        decimal += (binario % 10) * Math.pow(2, potencia);
        binario = binario / 10;
        potencia++;
    }
    return decimal;
}
```

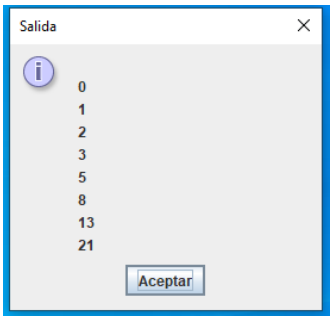




Fibonacci: toma un parámetro entero "contador". El método devuelve un String que representa la secuencia de Fibonacci hasta el valor de "contador" dado.

```java
public static String fibonacci(int contador)
{
    int Fib1 =0;
    int Fib2 =1;
    String cad="";
    int numero = 0;
    for (int i = 1; i <= contador; i++) {
        cad+= "\n"+numero;
        numero = Fib1 + Fib2;
        Fib1 = Fib2;
        Fib2 = numero;
    }
    return cad;
}
```

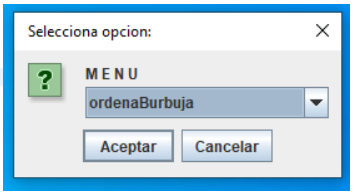


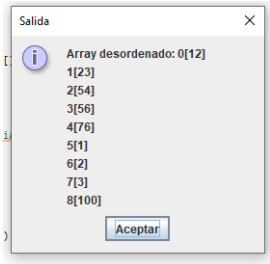


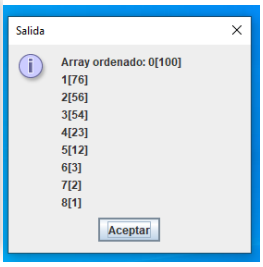
Ordena Burbuja:

comparando cada par de elementos auxiliares en el arreglo y cambiándolos de lugar si están en el orden incorrecto (ya sea ascendente o descendente). Este proceso se repite varias veces hasta que el arreglo esté completamente ordenado.

```
public static void ordenaBurbuja(int datos[])
{
    int i ,j, aux;

    for (i=0;i<datos.length-1;i++)
        for(j=i+1;j<datos.length;j++) {
            if (datos[i]<datos[j]) {//canbia el orden mayor o menor
                aux=datos[i];
                datos[i]=datos[j];
                datos[j]=aux;
            }
        }
  }
```







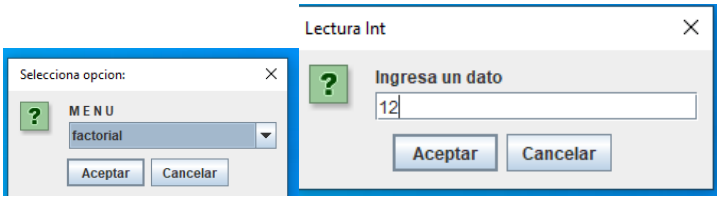
Ver Array:

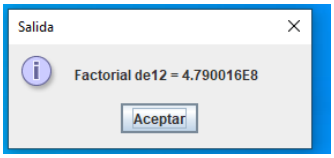
este método se usa para crear una representación de cadena con formato de una matriz de enteros, con el índice y el valor de cada elemento que se muestra en una línea separada.

```
public static String verArray (int datos[])
{
    String cad="";

    for (int k=0;k<datos.length;k++) {
        cad+=k+"["+datos[k]+"]"+"\n";
    }
    return cad;
}
```

Factorial: Este método calcula el factorial de un número entero dado utilizando un enfoque iterativo. Multiplica el valor del entero por todos los enteros positivos anteriores hasta llegar a 1, lo que da como resultado el valor factorial.

```
public static double factorial(int dato)
{
    int f=1,c=1;
    while (c<=dato)
    {
        f*=c;
        c++;
    }

    return f;

}
```





Leer Matriz:

este método se usa para llenar una matriz de enteros bidimensional con la entrada del usuario.

```
public static void leerMatriz(int a[][])
{
    int i,j;
    String ca="";
    for( i=0; i<a.length;i++)
    for( j=0; j<a[0].length;j++)
    {
     a[i] [j]=Integer.parseInt(JOptionPane.showInputDialog("Dato:"));
    }


}
```

Menú recursivo

Función Iterativa 1: esta función devuelve una cadena que enumera los valores de n0 a 10, cada uno con la palabra "incremento" delante, separados por caracteres de nueva línea.

```
public static String funcionIterativa(int n)
{
    return (n > 10)
            ? ""
            : "incremento" + n + "\n" + funcionIterativa(n + 1);

}
```

Función iterativa 2: Esta es una función recursiva en Java que toma dos parámetros enteros, j y n, y devuelve una cadena.

```
public static String funcionIterativa2(int j, int n)
{
    return (j <= n)
            ? "\nIncremento: " + j + funcionIterativa2(j + 1, n)
            : "";
}
```

Función iterativa 3: función recursiva que imprime una cadena de números crecientes desde j hasta n, cada uno separado por un carácter de nueva línea.

```
public static String funcionIterativa3(int j, int n)
{
    return(j<=n)? "\nIncremento"+j+funcionIterativa3(j+1,n):"";
}
```

Tabla de multiplicar forma recursiva: Este es un método Java que genera una tabla de multiplicar para un número dado num. El método toma dos argumentos enteros: num, que es el número para el que se genera la tabla de multiplicar, y j, que es el factor actual que se multiplica por num.

```
public static String tablaMul(int num, int j)
{
    if(j<=10)
    {
        return +num+"*"+"="+(num*j)+"\n"+tablaMul(num,(j+1));
    }
    else
        return "";
}
```

Array Recursivo: toma dos argumentos: una matriz de enteros "matriz" y un entero "índice". Devuelve una representación de cadena de los elementos en la matriz usando recursividad.

```
public static String ArrayRecursivo(int[] array, int index) {
    return (index >= array.length)
        ? ""
        : index + "[" + array[index] + "]\n"  + ArrayRecursivo(array, index + 1);
}
```

Suma Divisiones: la función calcula la suma de todos los divisores de "dato" que son menores que "dato". Lo hace comprobando recursivamente cada número del 1 al "dato" - 1 para ver si es un divisor de "dato". Si un número es divisor, lo suma a

la variable "suma". La función termina cuando "k" es mayor o igual que "dato". En ese momento, se devuelve la suma final.

```
public static int SumaDivi(int dato,int k)
{
    int suma=0;
    if(k<dato){
        if(dato % k==0)
            suma+=k;
        return suma+SumaDivi(dato,(int)(k+1));
    }
    return suma;
}
```

Decimal a octal: esta función convierte un número decimal en un número octal mediante recursividad y devuelve el número octal como un número entero.

```
public static int deciOctal(int nOctal,int cont, int decimal)
{
    if(decimal!=0)
        return deciOctal((nOctal + ((decimal % 8) * cont)), cont * 10, decimal / 8);
    return nOctal;
}
```

Potencia: potencia de un dígito dado. Toma dos parámetros: "digit" y "pow". "dígito" es el número base y "pow" es el exponente al que se eleva el número base.

```
public static int potencias(int digit, int pow) {
    return (pow == 0)
        ? 1
        : digit * potencias(digit, pow - 1);
}
```

Fibonacci: La función comienza comprobando si la iteración actual "i" es menor o igual que el número deseado de números de Fibonacci "contador". Si es así, devolverá una cadena que consiste en el número de Fibonacci actual "numero" concatenado con el resultado de llamar a la función recursivamente con los parámetros actualizados de la siguiente manera: nfib1 se convierte en nfib2, nfib2 se convierte en la suma de nfib1 y nfib2 (el siguiente Fibonacci número en la secuencia), y número se convierte en la suma de nfib1 y nfib2.

```
public static String fibonacci(int contador, int nfib1, int nfib2, int numero, int i)
  {
      if(i<=contador)
          return "\n"+ numero+fibonacci(contador, nfib2, (nfib2+nfib1), (nfib1+nfib2), i+1);
      else return "";
  }
```

Valor factorial: este método es calcular el factorial del número dado "numero" usando recursividad. El método primero verifica si el valor de "numero" es menor o igual a 20. Si lo es, calcula el factorial del número actual y lo agrega a la cadena de resultados junto con el número actual. Luego vuelve a llamar al método "ValorFact" con el siguiente número y el factorial actualizado.

```
//1 al 20 valor en factorial
public static String ValorFact(int numero, long factorial) {
    return (numero <=  20)
        ? numero + " = " + factorial + "\n" + ValorFact(numero + 1, factorial * (numero + 1))
        : "";
}
```

Factorial Recursivo: El método toma dos parámetros, "dato" y "c", ambos enteros. "dato" es el número para el que queremos calcular el factorial, mientras que "c" es una variable auxiliar que realiza un seguimiento del paso actual en el cálculo.

Lo primero que hace el método es verificar si "dato" es igual a 0 o 1. Si lo es, entonces el factorial es simplemente 1, por lo que el método devuelve 1.

Si "dato" no es igual a 0 o 1, entonces el método ingresa al bloque else. Aquí comprueba si "c" es menor o igual que "dato". Si es así, calcula el factorial recursivamente llamando al mismo método con "dato" y "c+1" como argumentos, y multiplica el resultado por "c". Esto continúa hasta que "c" se vuelve mayor que "dato".

Una vez que "c" se vuelve mayor que "dato", el método devuelve 1.

En general, este método calcula el factorial de un número usando la recursividad.

```
public static double factorialRecur(int dato , int c)
{
    if (dato ==0|| dato==1 )return 1;

else
    if(c<=dato)
    return c*factorialRecur(dato, c+1);
    else return 1;
}
```

Recursos materiales y equipo.

- Computadora.
- Software IDE Eclipse For Java Developers,
- Lectura de los materiales de apoyo tema 1.
- Notas de clases.

Conclusiones.

En conclusión, la recursividad es una técnica poderosa en la programación que permite la resolución eficiente de problemas al dividirlos en subproblemas más pequeños y similares al problema original. Los procedimientos recursivos son la implementación de esta técnica, y su uso puede mejorar la legibilidad y mantenibilidad del código.

A través de este informe de prácticas, se han presentado diferentes ejemplos y estrategias para trabajar con la recursividad, así como herramientas para analizar su complejidad temporal. También se han discutido las ventajas y desventajas de su uso, y se ha presentado información para ayudar a los programadores a aplicarla de manera efectiva.

En resumen, la recursividad y los procedimientos recursivos son herramientas valiosas en la programación, y su comprensión y aplicación adecuadas pueden mejorar significativamente la calidad y eficiencia del código.

Referencias.

- (S/f). Recuperado el 16 de abril de 2023, de http://file:///C:/Users/HP/Downloads/Dialnet-TresAplicacionesPracticasParaEntenderLaRecursivida-8180672%20(2).pdf
- KathleenDollard. (s/f). *Procedimientos recursivos - Visual Basic*. Microsoft.com. Recuperado el 22 de abril de 2023, de https://learn.microsoft.com/es-es/dotnet/visual-basic/programming-guide/language-features/procedures/recursive-procedures

- Lenguajes y Paradigmas de Programación, curso 2022-23 © Departamento Ciencia de la Computación e Inteligencia Artificial, Universidad de Alicante Domingo Gallardo, Cristina Pomares, Antonio Botía, Francisco Martínez
- Mancuzo, G. (2021, junio 25). *Ciclo de vida iterativo: qué es, sus ventajas y desventajas*. Blog - ComparaSoftware; ComparaSoftware. https://blog.comparasoftware.com/ciclo-de-vida-iterativo-que-es-y-cuales-son-sus-ventajas/