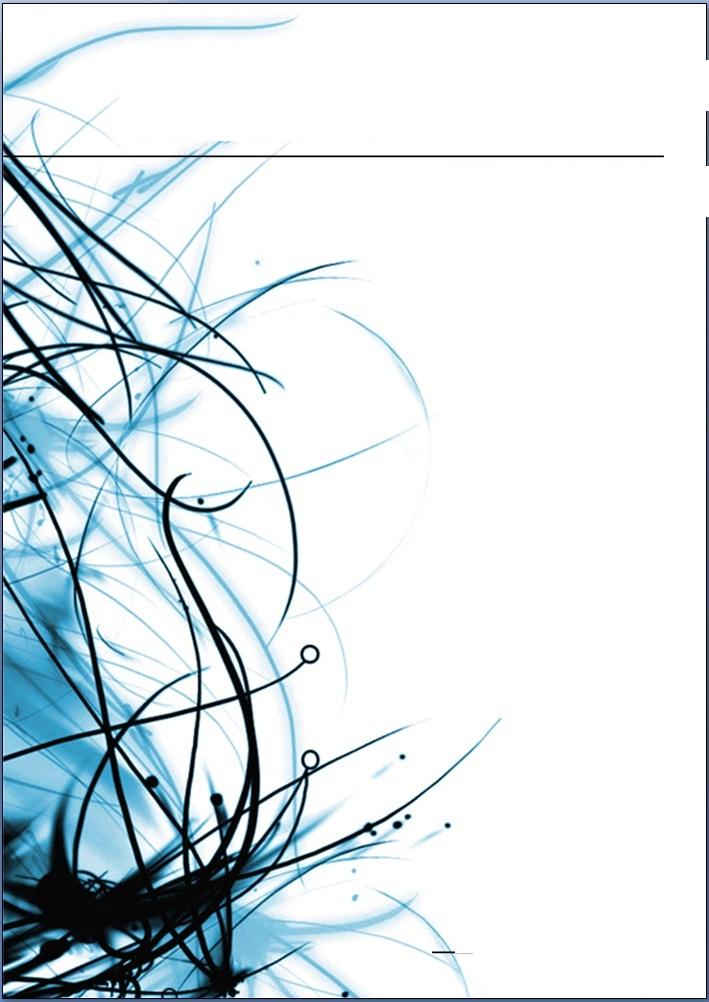
********Practico

Ciclos iterativos y recursivos

Grupo # 12

**Integrantes:**

Escalante Ustariz Eddy

Carita Paniora Joel David

Garcia Romero David

**Docente :**

Ing. Vargas Edwin Yapura

**Materia :**

INF-318 (Programacion logica funcional)

**Fecha :**

24/05/14

**Santa Cruz – Bolivia**

Introducción

ARITMÉTICA

• Se usa el predicado is

?- X is 3+4

X=7 yes.

• Uso de operaciones aritméticas en

predicados:

Suma(A, B, C):- C is A + B.

?- suma(3, 4, 7). yes.

?- suma(3, 4, X). X=7

yes.

CICLOS

• En Prolog, casi no se usan ciclos, en

Lugar de ellos se aplica recursividad; sin embargo, se pueden implementar.

• P. ejem. Para imprimir los números del 1

Al 10 se usa…

CICLO (M, N):- M<N, nl, write(M), NuevoM is M+1, CICLO (NuevoM, N).

1. **Preguntas**
2. sumaEnteros(N,Sum): predicado que encuentre en Sum la suma de los primeros n numeros enteros positivos. Ejemplo n=4, Sum=1+2+3+4.
3. sumaPares(N,Sum): predicado que encuentre en Sum la suma de los primeros n numeros pares. Ejemplo n=4, Sum=2+4+6+8.
4. sumaImpares(N,Sum): predicado que encuentre en Sum la suma de los primeros n numeros Impares. Ejemplo n=4, Sum=1+3+5+7.
5. factorial(N,F):predicado que encuentra en F el factorial del entero positivo N.
6. combi(N,R,NR):Predicado que encuentre en NR el numero combinatirio de N elementos tomados de R en R.
7. sumaCoef(N,Sum):predicados que necuentra en sum la suma de los coeficientes binomiales
8. potencia(X,N):predicado que en cuentre en pot,la potencia de X elevado a N ,X y N son valores enteros positivos.
9. sumPot(X,N,Sum):predicado que encuentra en Sum la suma de potencia
10. mostrarTabla(N):predicado que encuentra la tabvla multiplicar desde uno hasta n
11. mostrarFac(N):predicado que muestra pares de factores que multiplicados sean igual a N.
12. SumaInter(N,Sum):Predicados que encuentren en um la suma de los primeros N numeros positivos con signo intercalado
13. SumaGeon(N,Sum):Predicado que encuentren en sum la suma geometrica de los primeros N terminos
14. SumArmonica(N,Sum):Predicado que encuentren en sum la suma armonica de los primeros N terminos
15. SumAlterna(N,Sum):Predicado que encuentre en Sum la sumatoria de los primeros N terminos alternos
16. SumCuatrados(N,Sum): Predicado que encuentre en Sum la sumatoria de los primeros N terminos al cuadrado
17. SumEscalada(N,Sum): Predicado que encuentre en Sum la sumatoria escalada de sumatorias
18. SumaPiCuartos(N,Sum): Predicado que encuentre en Sum la sumatoria de los primeros N terminos que encuentre el equivalente de PI cuartos
19. SumaProd(X,N,Sum): Predicado que encuentre en Sum el exponenter a X
20. SumExpon(X,N,Sum): Predicado que encuentre en Sum el exponente a X
21. SumaSeno(X,N,Sum): Predicado que encuentre en Sum la sumatoria para encontrar el seno de X
22. SumaCoseno(X,N,Sum): Predicado que encuentre en Sum la sumatoria para encontrar el coseno de X

**Conclucion**

Los Ejercicios mencionados anteriormente ayudaron a nuestra manera de pensar desarrollando nuestra lógica para entender mejor el conocimiento que teníamos sobre las estructuras Cíclicas y verlo de una forma.

**Bibliografia**

Todo el conocimiento que hemos necesitado para resolver dicho practico fue obtenido durante la clase de Prolog (INF-318), puesto que no hemos utilizado algún otro conocimiento que no haiga sido enseñado por el docente de la materia

**Anexos**

**Codigo Iterativo y Recursivo (Java)**

package Ciclos;

public class Prolog\_Ciclos {

// FUNCIONES AUXILIARES

public boolean esPrimo(int n){

for (int i = 2; i <= n/2; i++) {

if (n % i == 0) return false;

} return true;

}

public boolean EsPerfecto(int n) {

int sum = 0;

for (int i = 1; i < n; i++) {

if (n % i == 0) {

sum += i;

}

}

return (sum == n);

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

// CICLICO -- CALCULA LA SUMA DE LOS PRIMEROS N NUMEROS ENTEROS

public int SumatoriaC(int n) {

int sum = 0;

for (int i = 1; i <= n; i++) {

sum += i;

}

return sum;

}

// RECURSIVO -- CALCULA LA SUMA DE LOS PRIMEROS N NUMEROS ENTEROS

public int SumatoriaR(int n) {

return SumatoriaR(n, 1);

}

private int SumatoriaR(int n, int i){ // METODO MASCARA

if (i > n) return 0;

return i + SumatoriaR(n, i+1);

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

// CICLICO -- CALCULA LA SUMA DE LOS PRIMEROS N NUMEROS PARES

public int SumaParesC(int n){

int c = 0, sum = 0;

for (int i = 2; c < n; i+=2) {

c++;

sum += i;

}

return sum;

}

// RECURSIVO -- CALCULA LA SUMA DE LOS PRIMEROS N NUMEROS PARES

public int SumaParesR(int n){

return SumaParesR(n, 2, 0);

}

private int SumaParesR(int n, int i, int c){

if (c >= n) return 0;

return i + SumaParesR(n, i+2, c+1);

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

// CICLICO -- CALCULA LA SUMA DE LOS PRIMEROS N NUMEROS PARES

public int SumaimParesC(int n){

int c = 0, sum = 0;

for (int i = 1; c < n; i+=2) {

c++;

sum += i;

}

return sum;

}

// RECURSIVO -- CALCULA LA SUMA DE LOS PRIMEROS N NUMEROS PARES

public int SumaimParesR(int n){

return SumaimParesR(n, 1, 0);

}

private int SumaimParesR(int n, int i, int c){

if (c >= n) return 0;

return i + SumaimParesR(n, i+2, c+1);

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

// CICLICO -- CALCULA EL FACTORIAL DE UN NUMERO

public int FactorialC(int n){

int r = 1;

for (int i = 1; i <= n; i++) {

r \*= i;

}

return r;

}

// RECURSIVO -- CALCULA EL FACTORIAL DE UN NUMERO

public int FactorialR(int n){

if (n == 0) return 1;

return n \* FactorialR(n-1);

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

// CICLICO -- CALCULA LA COMBINATORIA DE N EN R

public int CombiC(int n, int r){

return FactorialR(n) / (FactorialR(r) \* FactorialR(n-r));

}

// RECURSIVO -- CALCULA LA COMBINATORIA DE N EN R

public int CombiR(int n, int r){

return CombiC(n, r);

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

// CICLICO -- MOSTRAR COEFICIENTES DE N

public String MostrarCoefC(int n){

String r = "";

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

r += CombiC(n, i) + " ";

}

return r;

}

// RECURSIVO -- MOSTRAR COEFICIENTES DE N

public String MostrarCoefR(int n){

return MostrarCoefR(n, 0);

}

private String MostrarCoefR(int n, int i){

if (i > n) return "";

return CombiC(n, i) + " " + MostrarCoefR(n, i+1);

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

// CICLICO -- SUMAR COEFICIENTES DE N

public int SumarCoefC(int n){

int r = 0;

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

r += CombiC(n, i);

}

return r;

}

// RECURSIVO -- SUMAR COEFICIENTES DE N

public int SumarCoefR(int n){

return SumarCoefR(n, 0);

}

private int SumarCoefR(int n, int i){

if (i > n) return 0;

return CombiC(n, i) + SumarCoefR(n, i+1);

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

// CICLICO -- CALCULA LA POTENCIA DE X ELEVADO A N

public int PotenciaC(int x, int n){

int r = 1;

for (int i = 1; i <= n; i++) {

r \*= x;

}

return r;

}

// RECURSIVO -- CALCULA LA POTENCIA DE X ELEVADO A N

public int PotenciaR(int x, int n){

return PotenciaR(x, n, 1);

}

private int PotenciaR(int x, int n, int i){

if (i > n) return 1;

return x \* PotenciaR(x, n, i+1);

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

// CICLICO -- CALCULA LA SUMA DE LAS POTENCIAS DE X ELEVADO A I HASTA N

public int SumarPotenciaC(int x, int n){

int r = 0;

for (int i = 1; i <= n; i++) {

r += PotenciaC(x, i);

}

return r;

}

// RECURSIVO -- CALCULA LA SUMA DE LAS POTENCIAS DE X ELEVADO A I HASTA N

public int SumarPotenciaR(int x, int n){

return SumarPotenciaR(x, n, 1);

}

private int SumarPotenciaR(int x, int n, int i){

if (i > n) return 0;

return PotenciaR(x, i) + SumarPotenciaR(x, n, i+1);

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

// CICLICO -- MOSTRAR TABLA DE N

public void TablaC(int n){

for (int i = 1; i <= n; i++) {

for (int j = 1; j <= n; j++) {

System.out.println(i + " x " + j + " = " + (i\*j));

}

}

}

// RECURSIVO -- MOSTRAR TABLA DE N

public void TablaR(int n){

Tablai(n, 1);

}

private void Tablai(int n, int i){

if (i > n) return;

Tablaj(n, i, 1);

Tablai(n, i+1);

}

private void Tablaj(int n, int i, int j){

if (j > n) return;

System.out.println(i + " x " + j + " = " + (i\*j));

Tablaj(n, i, j+1);

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

// CICLICO -- MOSTRAR TABLA DE N

public void MostrarFactoresC(int n){

for (int i = 1; i <= n; i++) {

for (int j = 1; j <= n; j++) {

if (i \* j == n){

System.out.println(i + " x " + j + " = " + (i\*j));

}

}

}

}

// RECURSIVO -- MOSTRAR TABLA DE N

public void MostrarFactoresR(int n){

Factoresi(n, 1);

}

private void Factoresi(int n, int i){

if (i > n) return;

Factoresj(n, i, 1);

Factoresi(n, i+1);

}

private void Factoresj(int n, int i, int j){

if (j > n) return;

if (i \* j == n){

System.out.println(i + " x " + j + " = " + (i\*j));

}

Factoresj(n, i, j+1);

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

// CICLICO -- MOSTRAR PITAGORAS

public void MostrarPitagorasC(int n){

for (int i = 1; i <= n; i++) {

for (int j = 1; j <= n; j++) {

for (int k = 1; k <= n; k++) {

if (i\*i + j\*j == k\*k){

System.out.println(i + "^2 + " + j + "^2 = " + (i\*j));

}

}

}

}

}

// RECURSIVO -- MOSTRAR PITAGORAS

public void MostrarPitagorasR(int n){

pitagorai(n, 1);

}

private void pitagorai(int n, int i){

if (i > n) return;

pitagoraj(n, i, 1);

pitagorai(n, i+1);

}

private void pitagoraj(int n, int i, int j){

if (j > n) return;

pitagorak(n, i, j, 1);

pitagoraj(n, i, j+1);

}

private void pitagorak(int n, int i, int j, int k){

if (k > n) return;

if (i\*i + j\*j == k\*k){

System.out.println(i + "^2 + " + j + "^2 = " + (i\*j));

}

pitagorak(n, i, j, k+1);

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

// CICLICO -- MOSTRAR PRIMOS ENTRE A Y B

public String MostrarPrimosC(int a, int b) {

String r = "";

for (int i = a; i <= b; i++) {

if (esPrimo(i)) {

r += i + " ";

}

}

return r;

}

// RECURSIVO -- MOSTRAR PRIMOS ENTRE A Y B

public String MostrarPrimosR(int a, int b) {

return MostrarPrimos(b, a);

}

private String MostrarPrimos(int n, int i) {

if (i > n) return "";

if (esPrimo(i)) {

return i + " " + MostrarPrimos(n, i+1);

}

return MostrarPrimos(n, i+1);

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

// CICLICO -- MOSTRAR PERFECTOS ENTRE A Y B

public String MostrarPerfectosC(int a, int b) {

String r = "";

for (int i = a; i <= b; i++) {

if (EsPerfecto(i)) {

r += i + " ";

}

}

return r;

}

// RECURSIVO -- MOSTRAR PERFECTOS ENTRE A Y B

public String MostrarPerfectosR(int a, int b) {

return MostrarPerfectos(b, a);

}

private String MostrarPerfectos(int n, int i) {

if (i > n) return "";

if (EsPerfecto(i)) {

return i + " " + MostrarPerfectos(n, i+1);

}

return MostrarPerfectos(n, i+1);

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

// CICLICO -- MOSTRAR DIVISORES DE N

public String DivisoresC(int n) {

String r = "";

for (int i = 1; i <= n; i++) {

if (n % i == 0) {

r += i + " ";

}

}

return r;

}

// RECURSIVO -- MOSTRAR DIVISORES DE N

public String DivisoresR(int n) {

return Divisor(n, 1);

}

private String Divisor(int n, int i) {

if (i > n) return "";

if (n % i == 0) {

return i + " " + Divisor(n, i+1);

}

return Divisor(n, i+1);

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

// CICLICO -- SUMAR LOS DIVISORES DE N

public int SumarDivisoresc(int n) {

int sum = 0;

for (int i = 1; i <= n; i++) {

if (n % i == 0) {

sum += i;

}

}

return sum;

}

// RECURSIVO -- SUMAR LOS DIVISORES DE N

public int SumarDivisoresR(int n) {

return SumarDivisores(n, 1);

}

private int SumarDivisores(int n, int i){

if (i > n) return 0;

if (n % i == 0) return i + SumarDivisores(n, i+1);

return SumarDivisores(n, i+1);

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

public static void main(String[] args) {

Prolog\_Ciclos m = new Prolog\_Ciclos();

System.out.println("sumatoria ciclico " + m.SumatoriaC(5));

System.out.println("sumatoria recursivo " + m.SumatoriaR(5));

System.out.println("Suma pares ciclico " + m.SumaParesC(4));

System.out.println("Suma pares Recursivo " + m.SumaParesR(4));

System.out.println("Suma impares ciclico " + m.SumaimParesC(4));

System.out.println("Suma impares Recursivo " + m.SumaimParesR(4));

System.out.println("Factorial Ciclico " + m.FactorialC(5));

System.out.println("Factorial Recursivo " + m.FactorialR(5));

System.out.println("Combinatoria ciclica " + m.CombiC(5, 3));

System.out.println("Combinatoria recursiva " + m.CombiR(5, 3));

System.out.println("Mostrar Coeficientes ciclico " + m.MostrarCoefC(5));

System.out.println("Mostrar Coeficientes recursivo " + m.MostrarCoefR(5));

System.out.println("Sumar Coeficientes ciclico " + m.SumarCoefC(5));

System.out.println("Sumar Coeficientes recursivo " + m.SumarCoefR(5));

System.out.println("Potencia ciclico " + m.PotenciaC(2, 5));

System.out.println("Potencia recursivo " + m.PotenciaR(2, 5));

System.out.println("Sumar Potencias ciclico " + m.SumarPotenciaC(2, 5));

System.out.println("Sumar Potencias recursivo " + m.SumarPotenciaR(2, 5));

System.out.println("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*");

System.out.println("Tabla ciclica");

m.TablaC(5);

System.out.println("Tabla recursiva");

m.TablaR(5);

System.out.println("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*");

System.out.println("Mostrar Factores ciclica");

m.MostrarFactoresC(5);

System.out.println("Mostrar Factores recursiva");

m.MostrarFactoresR(5);

System.out.println("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*");

System.out.println("Mostrar Pitagoras ciclica");

m.MostrarPitagorasC(10);

System.out.println("Mostrar Pitagoras recursiva");

m.MostrarPitagorasR(10);

System.out.println("Mostrar primos ciclico " + m.MostrarPrimosC(1, 10));

System.out.println("Mostrar primos recursivo " + m.MostrarPrimosR(1, 10));

System.out.println("Mostrar perfectos ciclico " + m.MostrarPerfectosC(1, 10));

System.out.println("Mostrar perfectos recursivo " + m.MostrarPerfectosR(1, 10));

System.out.println("Mostrar divisores ciclico " + m.DivisoresC(10));

System.out.println("Mostrar divisores recursivo " + m.DivisoresR(10));

System.out.println("Sumar divisores ciclico " + m.SumarDivisoresc(10));

System.out.println("Sumar divisores recursivo " + m.SumarDivisoresR(10));

}

}

**Codigo Recursivo (Prolog)**

%% -----------------------------------------

sumaEnteros(N,Sum):-

sumaEnteros(N,Sum,0).

sumaEnteros(N,Sum,Ac):-1>N,Sum is Ac, !.

sumaEnteros(N,Sum,Ac):-Ac1 is (Ac + N),

N1 is N -1,

sumaEnteros(N1,Sum,Ac1).

%% -----------------------------------------

sumaPares(N,Sum):-

sumaPares(N,Sum,0).

sumaPares(N,Sum,Ac):-1>N,Sum is Ac,!.

sumaPares(N,Sum,Ac):-Ac1 is (Ac + (N\*2)),

N1 is N-1,

sumaPares(N1,Sum,Ac1).

%% -----------------------------------------

sumaImpares(N,Sum):-

sumaImpares(N,Sum,0).

sumaImpares(N,Sum,Ac):-1>N,Sum is Ac,!.

sumaImpares(N,Sum,Ac):-Ac1 is (Ac + ((N\*2)-1)),

N1 is N-1,

sumaImpares(N1,Sum,Ac1).

%% -----------------------------------------

factorial(N,F):-

N =\= 0,

factorial(N,F,1).

factorial(\_,F):-F is 1.

factorial(N,F,Ac):-1>N,F is Ac, !.

factorial(N,F,Ac):-Ac1 is (N\*Ac),

N1 is N-1,

factorial(N1,F,Ac1).

%% -----------------------------------------

combi(N,R,NR):-

T is (N-R),

factorial(T,T1),

factorial(N,N1),

factorial(R,R1),

P is (R1 \* T1),

NR is (N1/P).

%% -----------------------------------------

potencia(X,N,Pot):-

Pot is (X\*\*N).

sumPotencia(X,N,Sum):-

sumaPotencia(X,N,Sum,0).

sumaPotencia(\_,N,Sum,Ac):-1>N,Sum is Ac,!.

sumaPotencia(X,N,Sum,Ac):-

potencia(X,N,SP),

Ac1 is (Ac + SP),

N1 is N-1,

sumaPotencia(X,N1,Sum,Ac1).

%% -----------------------------------------

tabla(N):-

tabla(1,N).

tabla(I,N):-I>N,!.

tabla(I,N):-

tablaM(1,N,I),

I1 is I +1,

tabla(I1,N).

tablaM(I,N,\_):-I>N,!.

tablaM(I,N,M):-

M1 is (M\*I),

write(M),write(' \* '),write(I),write(' := '),write(M1),nl,

I1 is I+1,

tablaM(I1,N,M).

%% -----------------------------------------

mostrarFactores(N):-

mostrarFactores(1,N).

mostrarFactores(I,N):-I>N,!.

mostrarFactores(I,N):-

tablaFactores(1,N,I),

I1 is I +1,

mostrarFactores(I1,N).

tablaFactores(I,N,\_):-I>N,!.

tablaFactores(I,N,M):-

M1 is (M\*I),

M1 =:= 12,

write(M),write(' \* '),write(I),write(' := '),write(M1),nl,

I1 is I+1,

tablaFactores(I1,N,M).

tablaFactores(I,N,M):-

I1 is I+1,

tablaFactores(I1,N,M).

%% -----------------------------------------

sumaInter(N,Sum):-

sumaInter(N,Sum,0).

sumaInter(N,Sum,Ac):-1>N,Sum is Ac, !.

sumaInter(N,Sum,Ac):-Ac1 is ((((-1)\*\*N)\*N)+Ac),

N1 is N -1,

sumaInter(N1,Sum,Ac1).

%% -----------------------------------------

sumaGeom(N,Sum):-

sumaGeom(N,Sum,0).

sumaGeom(N,Sum,Ac):-1>N,Sum is Ac, !.

sumaGeom(N,Sum,Ac):-Ac1 is ((1/(2\*\*N))+Ac),

N1 is N -1,

sumaGeom(N1,Sum,Ac1).

%% -----------------------------------------

sumaArmonica(N,Sum):-

sumaGeom(N,Sum,0).

sumaArmonica(N,Sum,Ac):-1>N,Sum is Ac, !.

sumaGeom(N,Sum,Ac):-Ac1 is ((1/N)+Ac),

N1 is N -1,

sumaGeom(N1,Sum,Ac1).

%% -----------------------------------------

sumaInterna(N,Sum):-

sumaInterna(N,Sum,0).

sumaInterna(N,Sum,Ac):-1>N,Sum is Ac, !.

sumaInterna(N,Sum,Ac):-Ac1 is (((((-1)\*\*(N+1)))\*(1/N))+Ac),

N1 is N -1,

sumaInterna(N1,Sum,Ac1).

%% -----------------------------------------

sumaCuadrados(N,Sum):-

sumaCuadrados(N,Sum,0).

sumaCuadrados(N,Sum,Ac):-1>N,Sum is Ac, !.

sumaCuadrados(N,Sum,Ac):-Ac1 is ((N\*\*2)+Ac),

N1 is N -1,

sumaCuadrados(N1,Sum,Ac1).

%% -----------------------------------------

sumaescalada(N,S):-sumai(N,S,1).

sumai(N,S,I):-I>N,S is 0,!.

sumai(N,S,I):-sumaj(I,SJ,1),

I1 is I+1,

sumai(N,SI,I1),

S is SJ+SI.

sumaj(N,S,J):-J>N,S is 0,!.

sumaj(N,S,J):-J1 is J+1,

sumaj(N,S1,J1),

S is J+S1.

%% -----------------------------------------

sumapicuartos(N,S):-sumapi(N,S,1).

sumapi(N,S,I):-I>N,S is 0,!.

sumapi(N,S,I):-I1 is I+1,

sumapi(N,S1,I1),

R is (-1)^(I+1) \* 1 // (2\*I -1),

S is R+S1.

%% -----------------------------------------

sumaprod(N,S):-sumi(N,S,1).

sumi(N,S,I):-I>N,S is 0,!.

sumi(N,S,I):-sumj(N,SJ,I),.

I1 is I+1,

sumi(N,SI,I1),

S is SJ+SI.

sumj(N,S,J):-J>N,S is 0,!.

sumj(N,S,J):-J1 is J+1,

sumj(N,S1,J1),

S is J+S1