



Gobierno del
CHACO

Ministerio
de la Producción y el Desarrollo
Económico Sostenible



INFORMATARIO



PROGRAMACIÓN ESTRUCTURADA EN JAVA



Temas de hoy

1

Introducción a la programación
estructurada

2

Control de flujos

1

Introducción a la programación estructurada

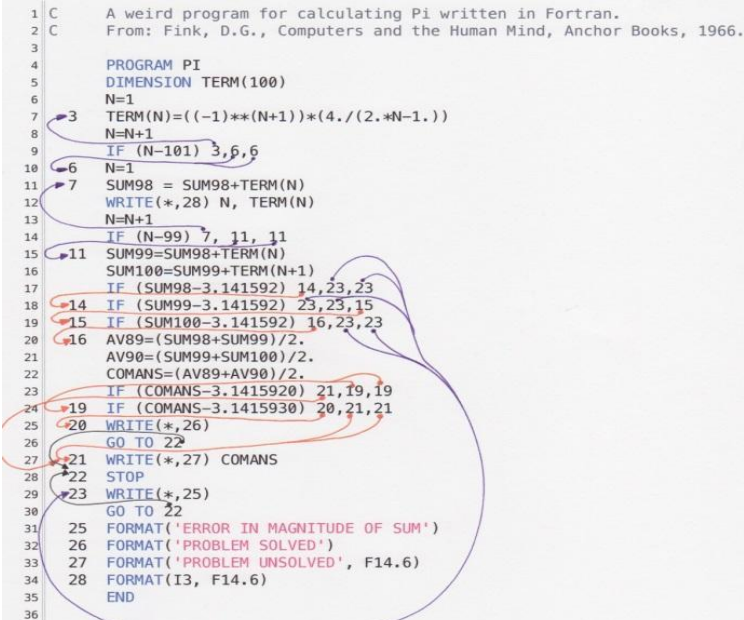
PROGRAMACIÓN ESTRUCTURADA

Todo lo programable es posible solo con condicionales, iterativas y secuenciales.

Dijkstra fue el héroe en este lío que exterminó el go-to (código espagueti) y permitió con la programación estructurada lograrlo todo..

PROGRAMACIÓN ESTRUCTURADA

```
1 C      A weird program for calculating Pi written in Fortran.
2 C      From: Fink, D.G., Computers and the Human Mind, Anchor Books, 1966.
3
4      PROGRAM PI
5      DIMENSION TERM(100)
6      N=1
7      TERM(N)=((-1)**(N+1))*(4./(2.*N-1.))
8      N=N+1
9      IF (N-101) 3,6,6
10     N=1
11     SUM98 = SUM98+TERM(N)
12     WRITE(*,28) N, TERM(N)
13     N=N+1
14     IF (N-99) 7, 11, 11
15     SUM99=SUM98+TERM(N)
16     SUM100=SUM99+TERM(N+1)
17     IF (SUM98-3.141592) 14,23,23
18     IF (SUM99-3.141592) 23,23,15
19     IF (SUM100-3.141592) 16,23,23
20     AV89=(SUM98+SUM99)/2.
21     AV90=(SUM99+SUM100)/2.
22     COMANS=(AV89+AV90)/2.
23     IF (COMANS-3.1415920) 21,19,19
24     IF (COMANS-3.1415930) 20,21,21
25     WRITE(*,26)
26     GO TO 22
27     WRITE(*,27) COMANS
28     STOP
29     WRITE(*,25)
30     GO TO 22
31     FORMAT('ERROR IN MAGNITUDE OF SUM')
32     FORMAT('PROBLEM SOLVED')
33     FORMAT('PROBLEM UNSOLVED', F14.6)
34     FORMAT(I3, F14.6)
35     END
36
```



The flowchart illustrates the control flow of the Fortran program. It starts at line 7, proceeds to line 8, then to line 9. From line 9, it branches to line 3 (labeled '3'), line 6 (labeled '6'), or line 6 (labeled '6'). From line 3, it goes to line 10. From line 6, it goes to line 11. From line 11, it goes to line 12. From line 12, it goes to line 13. From line 13, it branches to line 7 (labeled '7'), line 11 (labeled '11'), or line 11 (labeled '11'). From line 7, it goes to line 14. From line 11, it goes to line 15. From line 15, it branches to line 14 (labeled '14'), line 23 (labeled '23'), or line 23 (labeled '23'). From line 14, it goes to line 17. From line 23, it goes to line 18. From line 18, it branches to line 23 (labeled '23'), line 23 (labeled '23'), or line 15 (labeled '15'). From line 15, it goes to line 19. From line 19, it branches to line 16 (labeled '16'), line 23 (labeled '23'), or line 23 (labeled '23'). From line 16, it goes to line 20. From line 20, it goes to line 21. From line 21, it goes to line 22. From line 22, it branches to line 21 (labeled '21'), line 19 (labeled '19'), or line 19 (labeled '19'). From line 19, it goes to line 24. From line 24, it branches to line 20 (labeled '20'), line 21 (labeled '21'), or line 21 (labeled '21'). From line 20, it goes to line 25. From line 21, it goes to line 26. From line 26, it goes to line 27. From line 27, it goes to line 28. From line 28, it goes to line 29. From line 29, it goes to line 30. From line 30, it goes to line 31. From line 31, it goes to line 32. From line 32, it goes to line 33. From line 33, it goes to line 34. From line 34, it goes to line 35. From line 35, it goes to line 36.

PROGRAMACIÓN ESTRUCTURADA

Los lenguajes no estructurados como Fortran o BASIC hicieron de este tipo de mecanismo, y fueron el detonante de la necesidad de la **aparición de la programación estructurada** en la que se buscaba mejorar la claridad, calidad y tiempo de desarrollo de cualquier programa.

PROGRAMACIÓN ESTRUCTURADA

La programación estructurada es un paradigma de programación que se enfoca en la organización lógica y ordenada de un programa en módulos o bloques de código que realizan una tarea específica.

Se utilizan estructuras de control de flujo de ejecución como el **if, else, while y for, entre otros**. De esta manera, se fomenta la legibilidad y claridad del código, y se reducen los errores y la complejidad de los programas.

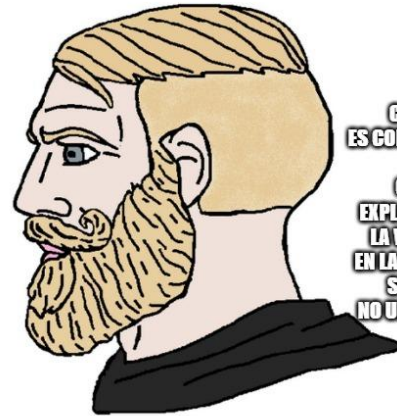
Stone Chad programador en Java vs. el arquitecto del caos



```
public void x() {  
    int y = 1;  
    if (y > 0) { while (true) { y++; if (y == 10) break;  
else { y--; } } }  
    System.out.println("Done");  
}
```


Stone Chad programador en Java vs. el arquitecto del caos

```
public void processUserData() {  
    int counter = 1;  
    while (counter < 10) {  
        counter++;  
    }  
    System.out.println("Processing  
complete.");  
}
```



**CÓDIGO DESORDENADO
ES COMO UNA BOMBA DE TIEMPO:
NO EXPLOTA
CUANDO LO ESCRIBÍS,
EXPLOTA CUANDO YA ES TARDE.
LA VERDADERA FUERZA ESTÁ
EN LA CLARIDAD, NO EN EL CAOS.
SE UN DESARROLLADOR,
NO UN ILUSIONISTA DEL ERROR.**

2

Control de flujos

CONTROL DE FLUJOS

- Simple
If (condición)
{ // Sentencias }
- Doble
If (condicion)
{ // Sentencias }
else { // Sentencias }

CONTROL DE FLUJOS

- Múltiple

```
If ( condición )  
{ //Sentencias }  
else if ( condición )  
{ //Sentencias }  
else if ( condición )  
{ //Sentencias }  
else ←(opcional)  
{ //Sentencias }
```

CONTROL DE FLUJOS

- Múltiple

```
switch(value){  
    case x :  
        //Código para valor == x  
        break;  
    case y :  
        //Código para valor == y  
    default:  
        //Código para valor default  
}
```

CONTROL DE FLUJOS

TIPOS DE VALORES VÁLIDOS PARA SWITCH

byte, short, int, char

Byte, Short, Integer, Character

String

enum

Importante : No puedes usar long, float o double o booleano o cualquier o sus ***Wrapper Class***.

CONTROL DE FLUJOS - ESTRUCTURAS ITERATIVAS

- Pre-test - While

While (condición)
{ // Sentencias}

Ejecuta el ciclo de 0 a N veces

CONTROL DE FLUJOS - ESTRUCTURAS ITERATIVAS

- Post-test - Do - While

Do{//Sentencias}
while (condición)

Ejecuta el ciclo de 1 a N veces

CONTROL DE FLUJOS - ESTRUCTURAS ITERATIVAS

- Manejado por contador - For
for (índice declarado;condición;inc/dec de índice)
{ // Sentencias}

Ejecuta el ciclo N veces

PROGRAMACIÓN ESTRUCTURADA

Recuerda las convenciones de nombre

Para carpetas o package : Todo en minúscula y junto, por ejemplo:
soyunacarpeta

Para nombre de clases o archivos .java : CamelCase, por ejemplo :
SoyUnArchivo.java

Para nombres de variables : lowerCase, por ejemplo : **soyUnaVariable**

Para nombres de constantes: SNAKE_CREAM_CASE, por ejemplo :
SOY_UNA_CONSTANTE_USAME