FUNDAMENTOS DE LA PROGRAMACIÓN

PROGRAMADOR UNIVERSITARIO EN INFORMÁTICA

2022

FUNDAMENTOS DE LA PROGRAMACIÓN

EQUIPO CÁTEDRA

PROFESORA RESPONSABLE

- **ING. SILVINA UNZAGA**

  sunzaga@unse.edu.ar

  Clase: Jueves 17 a 19 hs.

JEFE DE TRABAJOS PRÁCTICOS

- **ING. CLAUDIA ÁVILA**

  cavila@unse.edu.ar

  Clase: Martes 17 a 20 hs

JEFE DE TRABAJOS PRÁCTICOS

- **ING. CARMEN SILVA**

  csilva@unse.edu.ar

  Clase: Lunes de 14 a 17 hs

FUNDAMENTOS DE LA PROGRAMACIÓN

BIBLIOGRAFÍA

Título	Autor(es)	Editorial	Año y Lugar de edición	Disponible en
Algoritmos, datos y programas	De Giusti, Armando	Prentice Hall	2001, Argentina	Biblioteca Dpto. Informática. F.C.E.y T.
Fundamentos de Programación	Joyanes Aguilar, Luis	McGraw-Hill	2003, España	Biblioteca Dpto. Informática. F.C.E.y T.

FUNDAMENTOS DE LA PROGRAMACIÓN

ETAPAS EN LA SOLUCIÓN DE PROBLEMAS CON COMPUTADORA

- ***Análisis del problema***
- ***Diseño del algoritmo***
- ***Codificación del algoritmos***
- ***Verificación***
- ***Documentación***
- ***Mantenimiento***

FUNDAMENTOS DE LA PROGRAMACIÓN

ETAPAS EN LA SOLUCIÓN DE PROBLEMAS CON COMPUTADORA

ANÁLISIS DEL PROBLEMA

El análisis consiste en una clara definición del problema, donde se contemple exactamente lo que debe hacer el programa y el resultado o solución deseada.

Dado que se busca una solución por computadora, se precisan especificaciones detalladas de entradas y salidas.

Para resolver un problema con un ordenador hay que disponer de:

1. **Datos de entrada**
2. **El tratamiento que se ha de realizar a dichos datos.**
3. **La información que se desea obtener como resultado.**
4. **La forma que debe presentarse la información.**

FUNDAMENTOS DE LA PROGRAMACIÓN

ETAPAS EN LA SOLUCIÓN DE PROBLEMAS CON COMPUTADORA

ANÁLISIS DEL PROBLEMA

Ejemplo

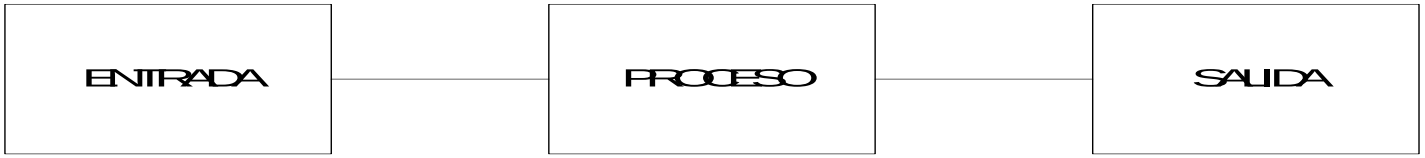
Se necesita hacer la nómina de los mejores alumnos de una cátedra se necesita saber:

ENTRADA: Los datos de cada uno de los alumnos y donde está toda la información de los alumnos.

PROCESO: La fórmula matemática para calcular el promedio de notas es:

(nota 1 + nota 2 + nota 3 + ...+ nota n) / cantidad de notas

SALIDA: El modelo del informe donde se desean imprimir el promedio de los alumnos.



FUNDAMENTOS DE LA PROGRAMACIÓN

ETAPAS EN LA SOLUCIÓN DE PROBLEMAS CON COMPUTADORA

DISEÑO DEL ALGORITMO

- Luego del análisis del problema, es preciso desarrollar un algoritmo que indique claramente los pasos a seguir para resolverlo. (un algoritmo es un método para resolver problemas)

- En esta etapa generalmente se utiliza el método “Divide y vencerás”, o “diseño descendente o modular”

- Para realizar un determinado proceso, el algoritmo debe ser independiente de la computadora en que resuelve el problema.

- Dada la importancia del algoritmo en la ciencia de la computación, un aspecto muy importante es el detalle del algoritmo.

- En esta etapa se realiza una representación de la secuencia, mediante: diagramas de flujo, pseudocódigos y/u otras formas.

FUNDAMENTOS DE LA PROGRAMACIÓN

ETAPAS EN LA SOLUCIÓN DE PROBLEMAS CON COMPUTADORA

CODIFICACIÓN DEL ALGORITMO

- Definido el algoritmo de resolución del problema, la siguiente etapa es generar el programa en cualquier lenguaje (C, pascal, Java, etc.) cuyo resultado será el programa fuente, el cual sigue las reglas de sintaxis del lenguaje escogido.
- Los programas se escriben con editores.
- El programa fuente deber ser traducido al único lenguaje que el ordenador comprende *"lenguaje de máquina".*
- Dicha operación se realiza mediante el correspondiente programa traductor o compilador del lenguaje en el que está escrito el programa.

FUNDAMENTOS DE LA PROGRAMACIÓN

ETAPAS EN LA SOLUCIÓN DE PROBLEMAS CON COMPUTADORA

VERIFICACIÓN

- Un algoritmo bien realizado y luego codificado en algún lenguaje de programación no significa que el programa resuelva correctamente el problema en cuestión.
- Para dar por finalizada cualquier labor de programación, es fundamental preparar un conjunto de datos lo más representativo posible del problema, que permitan probar el programa cuando se ejecute y así verificar los resultados.
- Cuanto más exhaustivas sean las pruebas de un programa, mayor seguridad se tendrá de que funcione correctamente y, por lo tanto, menor la posibilidad de errores.
- El programa se considera terminado cuando se han realizado pruebas y ensayos de su fiabilidad con conjuntos de datos reales.

ALGORITMO
Concepto

Un algoritmo es un conjunto finito de instrucciones que especifican una secuencia de operaciones que se deben realizar en orden, para resolver un problema específico o clase de problemas.

Un algoritmo es un método para la solución de un problema.

Definición

Un algoritmo puede definirse como una secuencia ordenada de pasos elementales, exenta de ambigüedades, que lleva a la solución de un problema dado en un tiempo finito.

PROPIEDADES DE LOS ALGORITMOS

1. **SECUENCIALIDAD**: debe haber una instrucción inicial y única y cada instrucción debe tener un sucesor único para un dato de entrada dado.
2. **AUSENCIA DE AMBIGÜEDAD**: Un algoritmo debe ser definido, claro, preciso y no ambiguo. Esta condición significa que cada vez que se presente para su ejecución un algoritmo con los mismos datos de entrada, se obtendrán los mismos resultados.
3. **GENERALIDAD**: Un algoritmo se puede utilizar para varios problemas que se relacionan entre si. Un algoritmo se aplica a un problema o clase de problemas específicos, el rango de las entradas o dominio se tiene que definir previamente, ya que este determina el alcance o la generalidad del algoritmo.
4. **LIMITACIÓN**: Un algoritmo es finito en tamaño y tiempo. La ejecución de un algoritmo programado debe finalizar después de que se haya llevado a cabo una cantidad finita de operaciones.

DOMINIO DE LOS ALGORITMOS

Se llama **DOMINIO** del algoritmo a la clase o el conjunto de datos y las condiciones para las cuales un algoritmo trabaja correctamente.

Cuando se trata de resolver cualquier problema es necesario definir el dominio del algoritmo y después verificar que trabaja para todos los casos que se encuentran dentro de ese dominio.

Al especificar el dominio del algoritmo, se define la **generalidad** del mismo.
Si un algoritmo no es general dentro de alguna clase de problemas entonces es de poca utilidad, siempre debe haber una restricción a la generalidad de un algoritmo.

Ejemplo:
Un algoritmo para marcar el número telefónico 4229058 podría no tiene valor para nadie, mientras que un algoritmo para marcar cualquier número podrá ser útil.

ERRORES EN LA COSTRUCCIÓN DE UN ALGORITMO

• **ERRORES DE DOMINIO:** Se presentan cuando no se han especificado todas las situaciones que se pueden presentar en la práctica o se ha descuidado la apreciación de su importancia.
A medida que el problema se presenta, se tiene que clasificar y hay tres opciones:

a. Ignorarlo porque es improbable y quizás nunca ocurra.
b. Restringir el dominio del algoritmo para excluirlo.
c. Corregir el algoritmo.

• **ERRORES DE LÓGICA:** Son aquellos errores que se detectan, después de que se ha definido en forma adecuada el dominio de un algoritmo, en la etapa de prueba o verificación.
Se deben principalmente a las siguientes causas:

a. Etapas incorrectas
b. Secuencia incorrecta de etapas.

FORMAS DE EXPRESAR UN ALGORITMO

Un mismo algoritmo puede ser expresado de distintas formas y en distintos lenguajes:

LENGUAJE COMÚN: en el lenguaje normal que hablamos y escribimos; útil para comunicar un algoritmo a otra persona o en una fase de análisis previo de un sistema computacional.

DIAGRAMAS DE FLUJO: es un lenguaje gráfico; útil para visualizar en forma rápida la secuencia lógica de pasos a seguir para un algoritmo y de gran ayuda para la traducción del mismo a un programa de computación.

TABLAS DE DECISIÓN: se usan para expresar en forma de tablas las distintas alternativas que intervienen en un problema y las operaciones elementales a realizar en cada alternativa. Muy útiles para analizar la lógica de un algoritmo en forma exhaustiva y precisa.

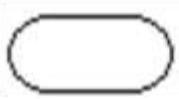
FORMAS DE EXPRESAR UN ALGORITMO *Continuación*

LENGUAJES DE PSEUDOCODIGOS: es un lenguaje intermedio entre nuestro lenguaje y un lenguaje de programación. El principal objetivo del pseudocódigo es el de representar la solución a un algoritmo de la forma más detallada posible y a su vez lo más parecida al lenguaje que posteriormente se utilizara para la codificación del mismo.

LENGUAJE DE PROGRAMACIÓN: es la forma obligada de expresión de un algoritmo para que pueda ser leído, ejecutado y almacenado por el computador. Está formado por un conjunto de instrucciones.

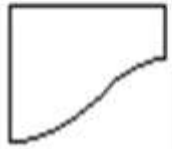
DIAGRAMAS DE FLUJO

Un **diagrama de flujo** es la representación gráfica o visual de un algoritmo. Se usan en el planeamiento, desarrollo y estructuración de un algoritmo. Mediante los diagramas de flujo el algoritmo se puede comunicar y documentar.



INICIO/FIN: Indica principio o fin de un diagrama de flujo. Puede usarse para indicar Pausa o Alto.

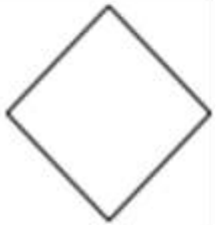
ENTRADA/LECTURA: Indica entrada de datos.



SALIDA IMPRESA de la información.

DIAGRAMAS DE FLUJO *continuación*

PROCESAMIENTO: Se usa generalmente para sentencias o enunciados de asignación (acción de asignar).



DECISIÓN: Se usa para evaluar expresiones de tipo lógica o aritmética. Si es de tipo lógica solamente dos salidas son posibles: V o F . Si la expresión es aritmética las salidas son tres >, < o = a.

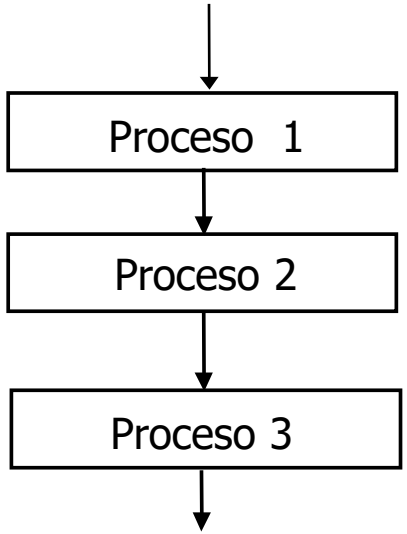
CONECTOR O SÍMBOLO DE CONEXIÓN: Se usa cuando el diagrama es largo y se requiere más de una hoja de papel o para evitar líneas que se crucen. Para cada rótulo de conector hay un único conector de entrada y puede haber más de uno de salida.

ESTRUCTURAS BÁSICAS DE CONTROL

- Estructura o lógica secuencial.
- Estructura o lógica de selección.
- Estructura o lógica de iteración o repetición.

ESTRUCTURA O LÓGICA SECUENCIAL

Diagrama de Flujo



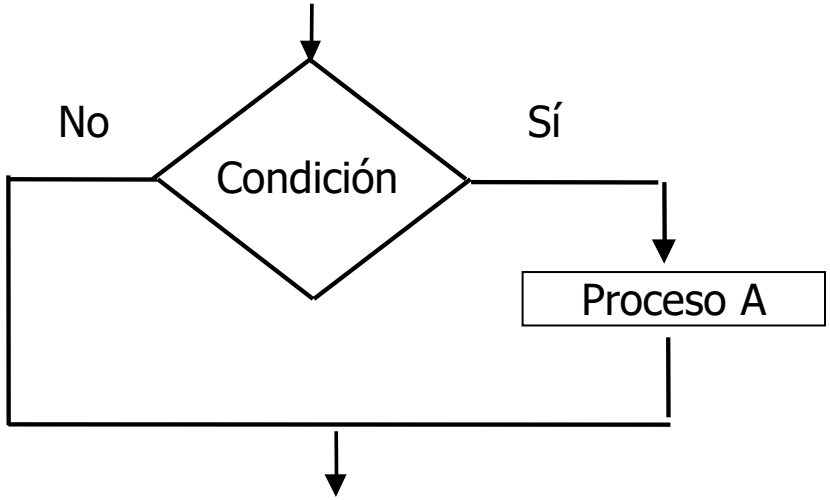
Pseudocódigo

```
. . .
Proceso 1
Proceso 2
Proceso 3
. . .
```

ESTRUCTURA DE SELECCIÓN

ALTERNATIVA SIMPLE

Diagrama de Flujo



Pseudocódigo

SI (Condición) ENTONCES

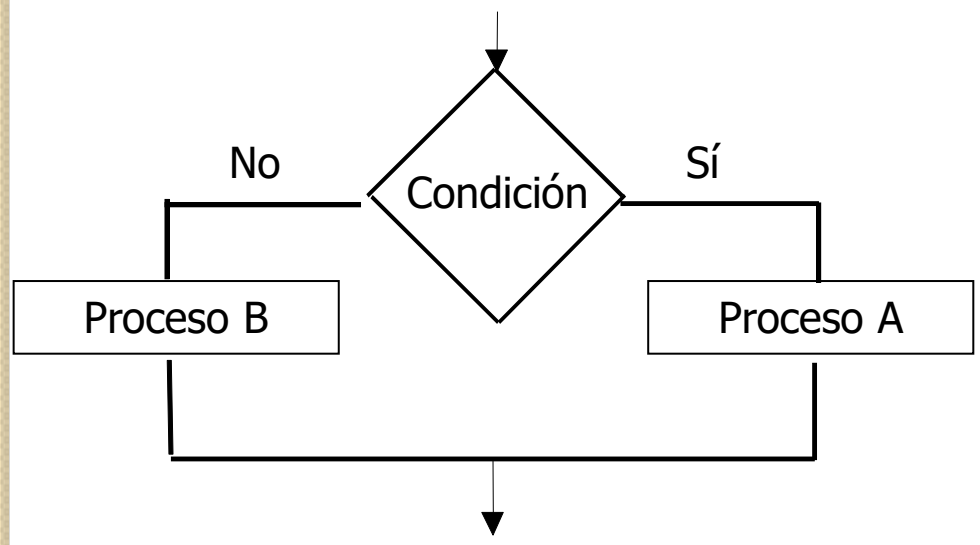
Proceso A

FINSI

ESTRUCTURA DE SELECCIÓN continuación

ALTERNATIVA DOBLE

Diagrama de Flujo



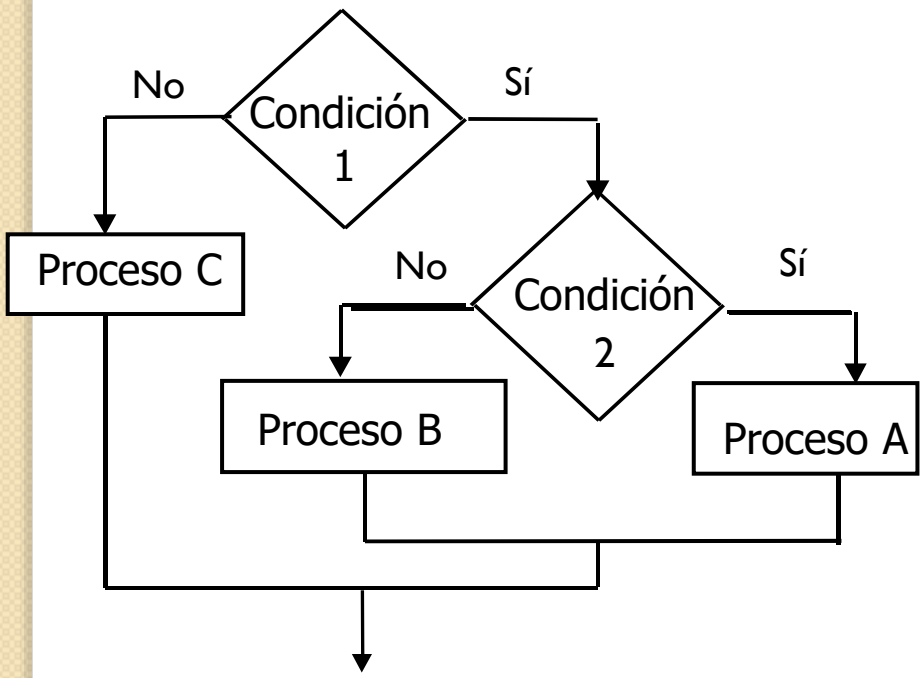
Pseudocódigo

```
SI (Condición ) ENTONCES
    Proceso A
SINO
    Proceso B
FINSI
```

ESTRUCTURA DE SELECCIÓN continuación

Alternativa múltiple

Diagrama de Flujo



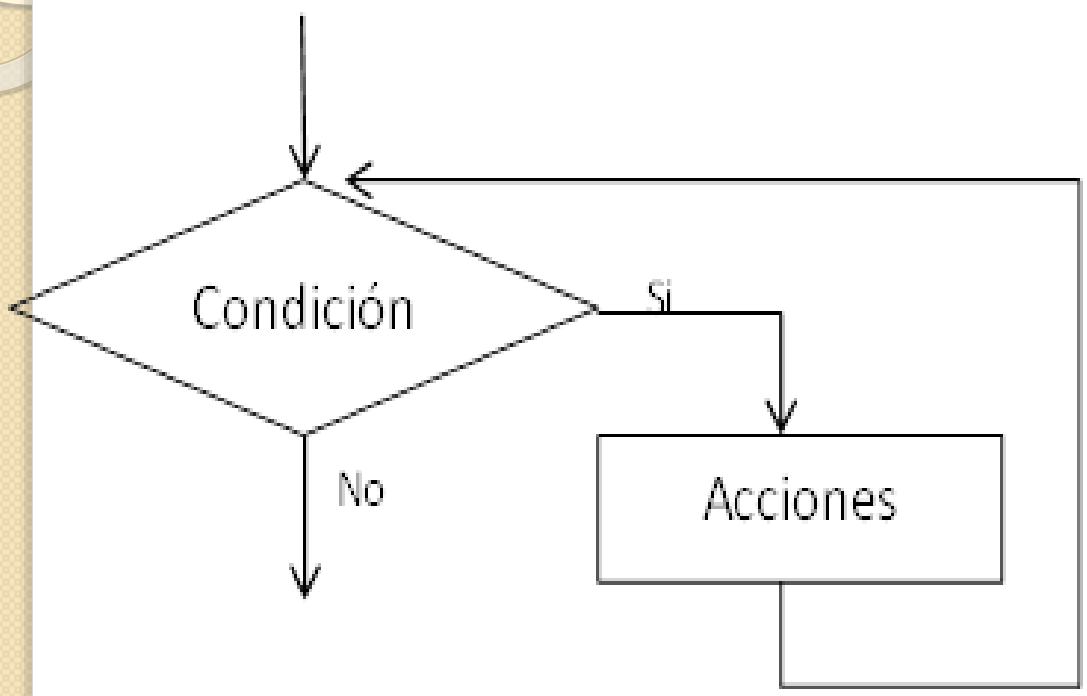
Pseudocódigo

```
SI (Condición 1) ENTONCES
    SI (Condición 2) ENTONCES
        Proceso A
    SINO
        Proceso B
    FINSI
SINO
    Proceso C
FINSI
```

ESTRUCTURA DE ITERACIÓN

Pre condicional- Do While (Hacer Mientras)

Diagrama de Flujo



Pseudocódigo

. . .

MIENTRAS (Condición)

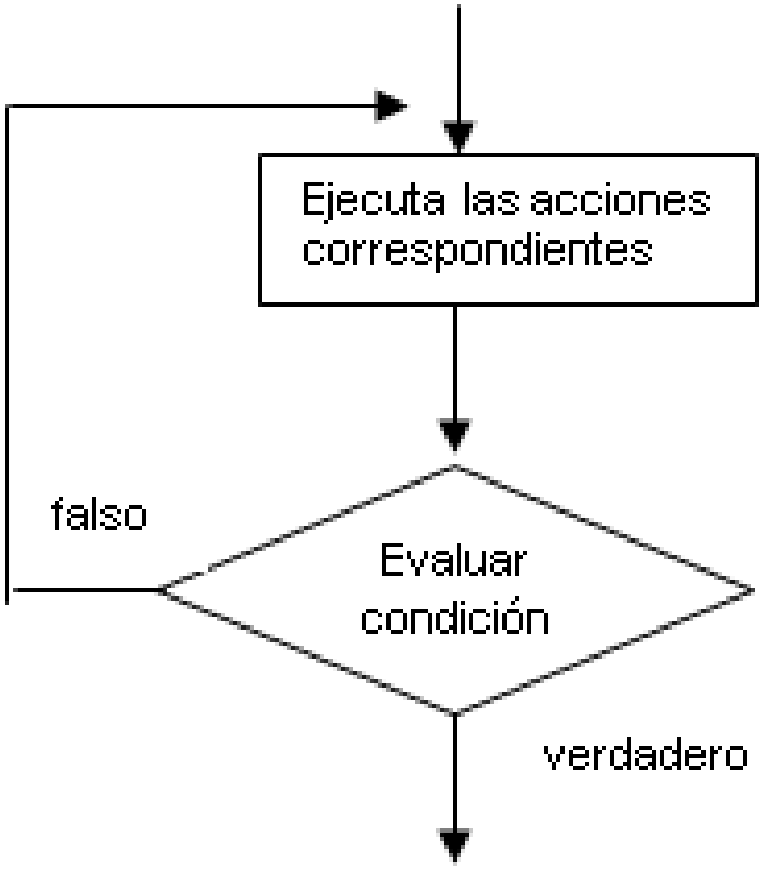
Acciones a repetir

FINMIENTRAS

ESTRUCTURA DE ITERACIÓN

Pos condicional- Repeat Until (Repetir Hasta)

Diagrama de Flujo



Pseudocódigo

. . .

REPETIR

Acciones a repetir

Hasta (Condición)

Problema: Leer **500** números enteros, calcular y mostrar la suma de los nros. positivos y la cantidad de los nros. negativos y nulos.

PSEUDOCÓDIGO

```
1.  Inicio
2.  SUMP = 0.
3.  CNEG = 0.
4.  CNU = 0.
5.  C = 1
6.  Mientras (C <= 500)
7.      Leer NUM
8.      Si (NUM >0 )
9.          Entonces
10.             SUMP = SUMP +NUM
11.         Sino
12.         Si (NUM < 0)
13.             Entonces
14.                 CNEG = CNEG + 1
15.             Sino
16.                 CNU = CNU + 1
17.         FinSi
18.     FinSi
19.     C = C + 1
20. FinMientras
21. Mostrar SUMP, CNEG, CNU
22. Fin
```

Problema: Leer **N** números enteros, calcular y mostrar la suma de los nros. positivos y la cantidad de los nros. negativos y nulos.

PSEUDOCÓDIGO

```
1.  Inicio
2.  SUMP = 0.
3.  CNEG = 0.
4.  CNU = 0.
5.  C = 1
6.  Leer N
7.  Mientras (C <=  N)
8.      Leer NUM
9.      Si (NUM >0 )
10.         Entonces
11.         SUMP = SUMP +NUM
12.         Sino
13.         Si (NUM < 0)
14.             Entonces
15.                 CNEG = CNEG +  1
16.             Sino
17.                 CNU = CNU + 1
18.         FinSi
19.     FinSi
20.     C = C + 1
21. FinMientras
22. Mostrar  SUMP, CNEG, CNU
23. Fin
```

Problema: Leer una serie de números enteros hasta ingresar el valor 999, calcular y mostrar la suma de los nros. positivos y la cantidad de los nros. negativos y nulos.

PSEUDOCÓDIGO

```
1.  Inicio
2.  SUMP = 0.
3.  CNEG = 0.
4.  CNU = 0.
5.  Leer NUM
6.  Mientras (NUM < > 999)
7.          Si (NUM >0 )
8.              Entonces
9.                  SUMP = SUMP +NUM
10.             Sino
11.             Si (NUM < 0)
12.                     Entonces
13.                         CNEG = CNEG +  1
14.                     Sino
15.                         CNU = CNU + 1
16.             FinSi
17.         FinSi
18.     Leer NUM
19. FinMientras
20. Mostrar  SUMP, CNEG, CNU
21. Fin
```

PSEUDOCÓDIGO
Estructura Post Condicional Repetir Hasta

```
1.  Inicio
2.  SUMP = 0.
3.  CNEG = 0.
4.  CNU = 0.
5.  C = 0
6.  Leer N
7.  Repetir
8.  Leer NUM
9.      Si (NUM >0 )
10.         Entonces
11.         SUMP = SUMP +NUM
12.         Sino
13.         Si (NUM < 0)
14.             Entonces
15.                 CNEG = CNEG +  1
16.             Sino
17.                 CNU = CNU + 1
18.         FinSi
19.     FinSi
20.     C = C + 1
21. Hasta (C = N )
22. Mostrar  SUMP, CNEG, CNU
23. Fin
```