Tema 4

Resolución de Problemas con Computadoras.

Algoritmos y Programas

TEMA 4

Etapas en la solución de problemas. Concepto de Algoritmo. Concepto de programa e instrucción. Técnicas y estructuras de programación.

Algoritmo y Programacion

Algoritmo: Es un método para resolver problemas que consiste en dividir el mismo en un número finito de pasos elementales e indicar claramente el orden de ejecución de los mismos.

Programación: Es la transformación del algoritmo en algo entendible por la computadora, para ello debe ser escrito en un lenguaje de programación (C, C++,PASCAL, BASIC, COBOL, ETC.) de acuerdo con las reglas de sintaxis del mismo.

4.1 Resolución de Problemas con Computadora

Las fases en la construcción de un programa para resolver un problema mediante la computadora son, en orden, las siguientes:

- **Análisis del problema**
- **Diseño del algoritmo**
- **Programación**
- **Ejecución y pruebas.**

El paso cero sería Entender el problema, parece banal, pero no lo es cuando se piensa en la gran cantidad de proyectos de computación que se desarrollaron sin haber comprendido bien para que se hacían, o cual era el problema que supuestamente iban a resolver. Y si además tomamos en cuenta que los sistemas de programación “reales”, a diferencia de los ejercicios de carácter didáctico o académicos, muchas veces son largos y complejos, e involucran la participación de varias personas durante largos períodos, podemos comprender la importancia de entender con claridad el problema antes de abocarnos a encontrar una solución.

4.1 Resolución de Problemas con Computadora

Análisis del problema
Diseño del algoritmo
Programación
Ejecución y pruebas.

El análisis consiste en estudiar el problema planteado para obtener una idea clara y concisa de los pasos necesarios para proponer un modelo para su solución. Es claro que este modelo no puede existir sin que se hayan especificado con claridad todos y cada uno de los componentes estructurales del problema. Nuestra función en esta etapa consiste precisamente en describir el modelo que mejor se adapte a la estructura del problema que estemos observando.

Para resolver un problema con un ordenador hay que disponer de los datos de entrada, estudiar el tratamiento que se ha de realizar a dichos datos, la información que se desea obtener como resultado y de que manera debe presentarse.

4.1 Resolución de Problemas con Computadora

Análisis del problema
Diseño del algoritmo
Programación
Ejecución y pruebas.

Es decir, después de analizar el problema, se han de conocer claramente tres cosas.

- *Datos de Entrada de que se dispone*
- *Proceso o Tratamiento que ha de realizarse con estos datos.*
- *Información de salida deseada.*

Una de las técnicas mas empleadas recibe el nombre de H.I.P.O. (Hierarchy the plus input process output) que consiste en esquematizar cada programa, o una parte del mismo en los tres bloques (los descritos anteriormente)..

4.1 Resolución de Problemas con Computadora

Análisis del problema
Diseño del algoritmo
Programación
Ejecución y pruebas.

Ejemplo

Sin entrar en el campo de la informática, para hacer la nómina de los mejores alumnos de una carrera, se necesita saber:

ENTRADA: Los datos de cada uno de los alumnos y si estos datos están en papel o en un fichero 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.

4.1 Resolución de Problemas con Computadora

Análisis del problema
Diseño del algoritmo
Programación
Ejecución y pruebas.

Teniendo en cuenta que un algoritmo es un método para resolver problemas, una vez analizado el mismo se precisa diseñar un algoritmo que indique claramente los pasos a seguir para resolverlo.

Para realizar un determinado proceso, se le debe suministrar al ordenador una fórmula para la resolución de un problema (*algoritmo*), cuyo diseño debe ser independiente de la computadora que resuelve el problema.

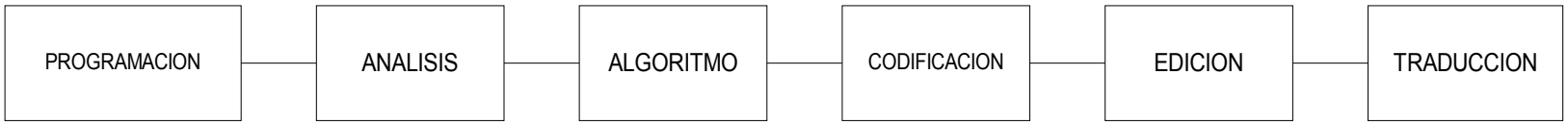
Dada la importancia del algoritmo en la ciencia de la computación, un aspecto muy importante será el *diseño del algoritmo.*

En esta etapa se realizará una representación gráfica de la secuencia. Estas representaciones son las herramientas pueden ser: *diagramas de flujo, pseudocódigos y/o tablas de decisión.*

4.1 Resolución de Problemas con Computadora

Análisis del problema
Diseño del algoritmo
Programación
Ejecución y pruebas.

Una vez que el diagrama de flujo o el algoritmo de resolución del problema está definido se pasa a la fase de codificación del programa en cualquier lenguaje (C, basic, cobol, pascal, etc.) cuyo resultado será el programa fuente el cual sigue las reglas de sintaxis que el lenguaje escogido exija.
Después de codificado el programa, se introduce en el ordenador mediante unos programas especiales llamados editores.
Una vez dentro del ordenador, el programa deber ser traducido al único lenguaje que éste entiende: Lenguaje de máquina. Dicha operación se realiza mediante el correspondiente programa traductor o compilador del lenguaje en el que está escrito el programa.



4.1 Resolución de Problemas con Computadora

Análisis del problema
Diseño del algoritmo
Programación
Ejecución y pruebas.

El hecho de haber diseñado un buen algoritmo y luego haberlo codificado en algún lenguaje de programación no significa que el programa resuelva correctamente el problema en cuestión.
Por eso, antes de 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 éste funcione correctamente y, por lo tanto, menor posibilidad de errores y por ende, mayor probabilidad habrá de evitar la tarea de revisar un antiguo programa, cuando ya la lógica que se empleó en el mismo se recuerda muy poco.
El programa se considera terminado cuando se han realizado pruebas y ensayo de su fiabilidad con el conjunto de datos seleccionados y otros nuevos, hasta incluso con datos reales, y no se encuentren errores de ningún tipo.

4.2 Concepto de Algoritmo

Algoritmo es un conjunto ordenado y finito de pasos que especifican la secuencia de operaciones que se han de realizar para resolver un problema.

Podemos entonces decir que *un algoritmo es un conjunto de reglas para resolver una cierta clase de problemas o una forma de describir la solución de un problema.* (Luis Joyanes)

Los algoritmos son independientes del lenguaje de programación en que se expresan como así también de la computadora que se ejecuten.

Un algoritmo se puede expresar en distintos lenguajes de programación y en computadoras distintas, pero el algoritmo, los pasos a seguir para la solución del problema es siempre el mismo.

Así como, cualquier cosa que ocurra en la vida cotidiana, por ejemplo, poner en movimiento un automóvil, tiene un número de pasos a seguir, sea quien sea el conductor: argentino, español, alemán, etc.- y sea cual sea el auto a conducir.

4.2 Concepto de Algoritmo

En la ciencia de la computación y específicamente en la programación, los algoritmos son más importantes que los lenguajes de programación e incluso que las computadoras, dado que los lenguajes de programación son solo un medio para expresar un algoritmo y las computadoras la herramienta que los ejecuta.

Debido a que la computadora es incapaz de tomar ninguna decisión propia sin que se especifique explícitamente, es imprescindible que el algoritmo elegido para resolver el problema sea absolutamente claro, sin ambigüedades y además contemple todas y cada una de las posibles situaciones que puedan presentarse durante la resolución del mismo.

En general, cualquier actividad de la vida cotidiana se puede describir mediante algoritmos. Para empezar a familiarizarnos con ellos, se desarrolla a continuación en ejemplo que pone de manifiesto la necesidad de las características antes mencionadas.

4.2 Concepto de Algoritmo

Ejemplo

Si a un experimentado conductor se le preguntase como pone en movimiento su automóvil, el contestaría: “Se pone en marcha el motor y se mete la primera”.

Efectivamente es fácil. Pero ¿Qué ocurriría si el individuo nunca condujo un automóvil?. El resultado no sería muy efectivo debido a que según las instrucciones o pasos impartidos anteriormente, si estuviera en algún cambio, al poner en marcha se hubiese estrellado con lo primero que estuviese adelante o atrás.

La conclusión que se debe sacar de este ejemplo es que el conductor con experiencia no tuvo en cuenta todas las posibilidades que se pueden presentar para obtener el resultado de poner el automóvil en movimiento. Por lo tanto, su algoritmo sería:

Poner en marcha el motor
Meter la primera

4.2 Concepto de Algoritmo

Ejemplo

En cambio, el algoritmo más correcto podría ser:

Pisar el embrague con el pie izquierdo
Poner en punto muerto
Dar a la llave de contacto
Pisar el embrague
Meter la primera
Quitar el freno de mano si lo tuviese puesto.
Levantar lentamente el pie del embrague a la vez que pisa el pedal del acelerador con el pie derecho

¿Por qué es más correcto este algoritmo?

1) desglosa el problema en instrucciones simples y concretas, comprensibles para cualquier individuo.

2) indica claramente el orden en que deben ejecutarse dichas instrucciones.

4.2 Concepto de Algoritmo

4.2.1 Caracteristicas

Se puede observar que el número de operaciones que realiza un algoritmo es finito siempre y cuando sus datos sean adecuados. Por consiguiente, el número de operaciones que necesitamos realizar al ejecutar un algoritmo dependerá de los datos del problema y solamente se conocerá al ejecutar este.

Un algoritmo debe ser:

Preciso: Debe indicar el orden de realización de cada paso.

Definido: Si se ejecuta dos veces el algoritmo con los mismos datos éste debe dar el mismo resultado.

Finito: Debe finalizar en algún momento o sea tener un número finito de pasos.

Todo algoritmo tiene tres partes: entrada, proceso y salida, y sus pasos describen la transformación de la entrada en la salida.

Si tomamos el ejemplo acerca del promedio de notas de los alumnos tenemos que:
ENTRADA: las notas de los alumnos
PROCESO: cálculo del promedio
SALIDA: Promedio de los alumnos

4.3 Programación

Para que un algoritmo pueda ser resuelto por una computadora el mismo debe ser escrito (codificado) en el lenguaje de programación elegido, siguiendo las reglas de sintaxis del mismo.

Esta tarea se denomina programación y el algoritmo escrito se llama programa.

4.3 Programación

4.3.1 Elementos básicos de un programa

Los elementos de programación son aquellos que permiten definir un lenguaje de comunicación con la computadora, y como todo lenguaje consta de:

El juego de caracteres [1,2...0,a,b.....z,*,-( )...]
Constantes
Variables
Operadores
Expresiones aritméticas

4.3 Programación

4.3.1. Elementos básicos de un programa

El juego de caracteres [1,2...0,a,b.....z,*,-( )...]
Constantes
Variables
Operadores
Expresiones aritméticas

Un carácter es un elemento pequeño utilizado en el tratamiento de la información. Un dato de tipo carácter contiene un solo carácter, siendo un *carácter* un conjunto finito y ordenado de caracteres que la computadora reconoce. Si bien estos caracteres no son estándar, la mayoría de las computadoras reconoce los caracteres alfabéticos, numéricos y especiales.

Alfabéticos (a,b,c,d, ...z) (A,B,C,D,.....Z)
Numéricos (0, 1, 2, 3, ... 9)
*Especiales (+, -, *, /, < , >, $.....)*

4.3 Programación

4.3.1 Elementos básicos de un programa

El juego de caracteres [1,2...0,a,b.....z,*,-( )...]
Constantes
Variables
Operadores
Expresiones aritméticas

Es el dato cuyo valor permanece inalterado durante los diferentes tratamientos, durante el desarrollo del algoritmo o en la ejecución de un programa.

3.1415 Constante numérica

A una secuencia de caracteres se la denomina *cadena* y si esta es una constante, se la encierra entre apóstrofos

‘Pedro Díaz
‘ 25 de Mayo’

Si dentro de la cadena hay apóstrofos como parte de la misma, se debe colocar un par de apóstrofos

‘ Maria’’s’

4.3 Programación

4.3.1 Elementos básicos de un programa

El juego de caracteres [1,2...0,a,b.....z,*,-( )...]
Constantes
Variables
Operadores
Expresiones aritméticas

Variable es un dato donde su valor puede ser modificado durante la ejecución del algoritmo o en un programa.

En informática, cuando hacemos mención a una variable, nos estamos refiriendo a una pequeña zona de la memoria principal donde se va a alojar un valor. Si este valor se modifica en algún momento del programa, el nuevo valor sustituirá al que existía anteriormente.

A este nombre de posiciones contiguas de memoria se le dan atributos: un nombre para poder referenciarlo (nombre de la variable) y su tipo (clase de caracteres que puede contener). A una variable definida de un determinado tipo, no se le puede asignar generalmente valores de otro tipo.

4.3 Programación

4.3.1 Elementos básicos de un programa

El juego de caracteres [1,2...0,a,b.....z,*,-( )...]
Constantes
Variables
Operadores
Expresiones aritméticas

El nombre que se le da a una variable lo elige el programador y se debe componer de caracteres alfanuméricos, generalmente se elige como primer carácter una letra. No se deben utilizar como nombre de variables palabras reservadas del lenguaje de programación.

Es aconsejable que el nombre que se le atribuya a la variable sea 'nemotécnico' o significativo, es decir su nombre debe guardar relación con el objeto que representa a fin de que la misma nos recuerde la naturaleza de la información que contiene. Algunos nombres de variables son:

NOTA *representa notas de alumnos*
NOMBRE_APELLIDO *representa el nombre y apellido de personas*
PRECIO *representa precios de artículos*

4.3 Programación

4.3.1 Elementos básicos de un programa

El juego de caracteres [1,2...0,a,b.....z,*,-( )...]
Constantes
Variables
Operadores
Expresiones aritméticas

Hay que diferenciar entre nombre de la variable y su contenido. El nombre es una identificación que se le da a un conjunto de posiciones contiguas de memoria, mientras que el contenido de una variable es el valor que está almacenado en dichas posiciones.

4.3 Programación

4.3.1 Elementos básicos de un programa

El juego de caracteres [1,2...0,a,b.....z,*,-( )...]
Constantes
Variables
Operadores
Expresiones aritméticas

Hay tres tipos de operadores:
aritméticos
relacionales
lógicos

4.3 Programación

4.3.1 Elementos básicos de un programa

El juego de caracteres [1,2...0,a,b.....z,*,-( )...]
Constantes
Variables
Operadores
Expresiones aritméticas

Hay tres tipos de operadores:

aritméticos
relacionales
lógicos

+	Suma
-	Resta
*	Multiplicación
/	División
** o ^	Potenciación

4.3 Programación

4.3.1 Elementos básicos de un programa

El juego de caracteres [1,2...0,a,b.....z,*,-( )...]
Constantes
Variables
Operadores
Expresiones aritméticas

Hay tres tipos de operadores:
aritméticos
relacionales
lógicos

=	Igual
<	Menor que
<=	Menor o igual que
>	Mayor
>=	Mayor o Igual que
<>	Distinto

4.3 Programación

4.3.1 Elementos básicos de un programa

El juego de caracteres [1,2...0,a,b.....z,*,-( )...]
Constantes
Variables
Operadores
Expresiones aritméticas

Hay tres tipos de operadores:
aritméticos
relacionales
lógicos

AND	Y
OR	O
NOT	NO

4.3 Programación

4.3.1. Elementos básicos de un programa

El juego de caracteres [1,2...0,a,b.....z,*,-( )...]
Constantes
Variables
Operadores
Expresiones aritméticas

Reglas de Prioridad

Las operaciones aritméticas siguen reglas de prioridad o precedencia y son:

operador exponencial *, ^

operadores de multiplicación y división, /

operadores de suma y resta +, -

4.3 Programación

3.2.1. Elementos básicos de un programa

El juego de caracteres [1,2...0,a,b.....z,*,-( )...]
Constantes
Variables
Operadores
Expresiones aritméticas

Las expresiones aritméticas son combinaciones de constantes, variables, símbolos de operación, paréntesis y nombres de funciones especiales. Las mismas son utilizadas en notación matemática tradicional.

a + (b – 20) * 2
(a + b) 2**

Cada expresión tiene un valor, que se determina tomando los valores de las variables y constantes implicadas y ejecutando las operaciones indicadas.

4.3 Programación

4.3.1. Elementos básicos de un programa

El juego de caracteres [1,2...0,a,b.....z,*,-( )...]
Constantes
Variables
Operadores
Expresiones aritméticas

Reglas de Prioridad

Las expresiones que tengan dos o más operadores requieren reglas matemáticas que permitan determinar el orden de las operaciones, dichas reglas son de prioridad o precedencia y son:

- Las operaciones que están encerradas entre paréntesis se evalúan primero. Si existen paréntesis anidados, los mismos se resuelven de adentro hacia fuera.
- Las operaciones aritméticas dentro de una expresión siguen el siguiente orden de prioridad:
- Operador exponencial
- Operadores * y / (multiplicación y división)
- Operadores + y - (suma y resta)

En el caso de coincidir operadores de igual jerarquía en una expresión encerrada entre paréntesis, el orden de prioridad se resuelve de izquierda a derecha.

4.3 Programación

4.3.1. Elementos básicos de un programa

El juego de caracteres [1,2...0,a,b.....z,*,-( )...]
Constantes
Variables
Operadores
Expresiones aritméticas

Ejemplo

((4-2) * (5 + 1) / 2) ** 2 – (4 + 3)
(2 * (5 + 1) / 2) ** 2 – (4 + 3)
(2 * 6 / 2) ** 2 – (4 + 3)
(12 / 2) ** 2 – (4 + 3)
6 ** 2 – (4 + 3)
6 ** 2 – 7
36 - 7
29

4.3 Programación

4.3.2 Instrucciones

Como ya se mencionó anteriormente, un algoritmo es un conjunto de acciones que se han de ejecutar para la resolución de un problema. A cada una de estas acciones se le denomina Instrucción o Sentencia.

Un conjunto de Instrucciones forma un programa. Las instrucciones se deben escribir y luego almacenar en memoria en el mismo orden en que han de ejecutarse, es decir, en secuencia.

Las instrucciones básicas que se pueden implementar en un algoritmo soportan todos los lenguajes de programación. Dicho de otro modo, las instrucciones básicas son independientes del lenguaje de programación.

La clasificación más corriente es:

Instrucciones de Comienzo y Fin

Instrucciones de transferencia

Instrucciones de entrada

Instrucciones de salida

Instrucciones de control

4.3.2 Instrucciones

Instrucciones de Comienzo y Fin
Instrucciones de transferencia
Instrucciones de entrada
Instrucciones de salida
Instrucciones de control

Son las instrucciones que ordenan el comienzo o fin del algoritmo.
Todo programa debe comenzar con la instrucción INICIO o COMENZAR o su simbología correspondiente
Todo programa debe finalizar con la instrucción FIN o PARAR o su simbología correspondiente.

4.3.2 Instrucciones

Instrucciones de Comienzo y Fin
Instrucciones de transferencia
Instrucciones de entrada
Instrucciones de salida
Instrucciones de control

Aritméticas

Cualquier operación aritmética que se desea realizar es llamada con este nombre.
Tienen dos etapas:
La ejecución de la operación, que implica la obtención de un resultado.
Una transferencia para almacenar en un campo de la memoria el resultado obtenido.
La forma general de la instrucción aritmética es:

b → a o a = b

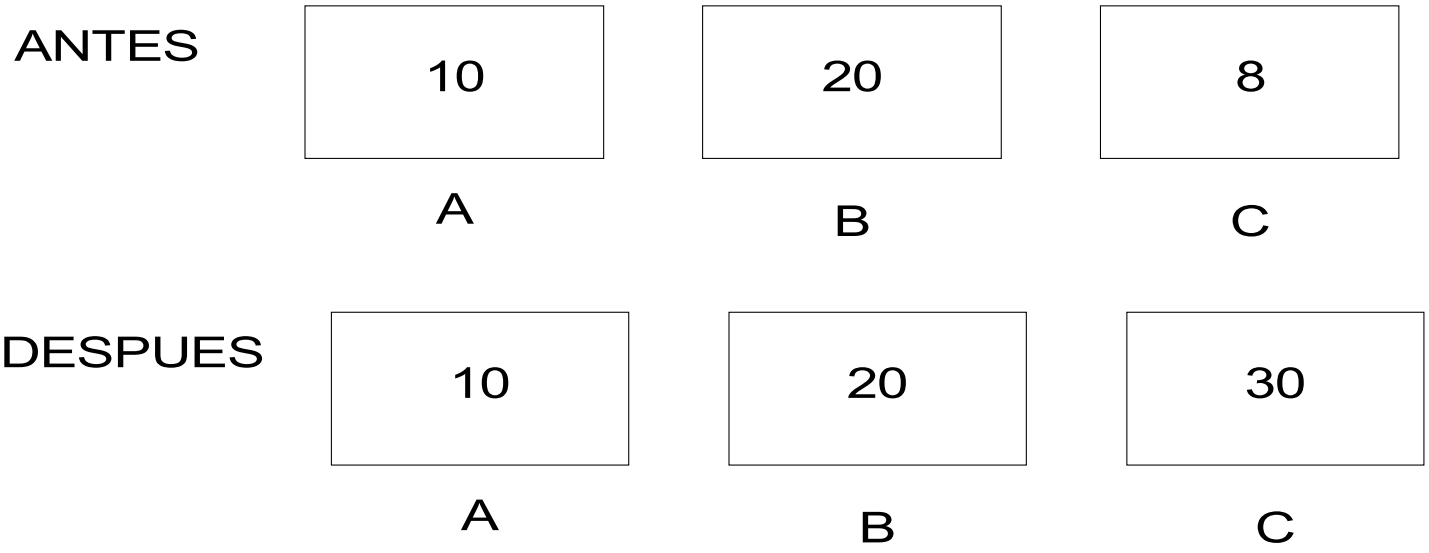
donde:
a es el nombre de una variable donde se almacena el resultado de b, y
b es una expresión aritmética

4.3.2 Instrucciones

Instrucciones de Comienzo y Fin
Instrucciones de transferencia
Instrucciones de entrada
Instrucciones de salida
Instrucciones de control

Aritméticas

Ejemplo: Sumar el contenido de los campos A y B
La instrucción sería (ver figura 1.16):
A + B → C o C = A + B

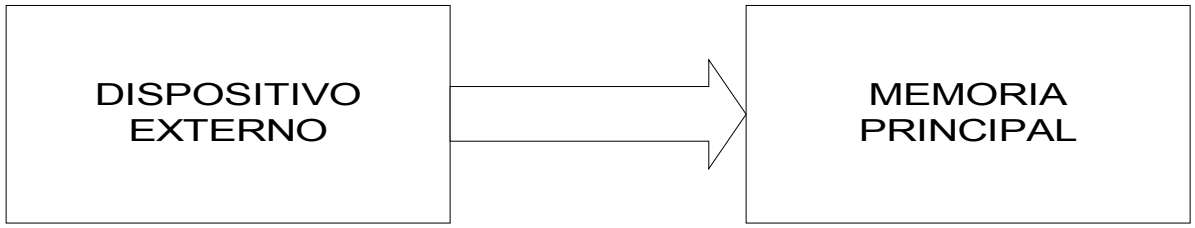


4.3.2 Instrucciones

Instrucciones de Comienzo y Fin
Instrucciones de transferencia
Instrucciones de entrada
Instrucciones de salida
Instrucciones de control

Esta instrucción introduce datos desde algún dispositivo de entrada. Una instrucción de Entrada implica la introducción de datos en la memoria principal del ordenador desde dispositivos externos a la misma, por ejemplo, el teclado, un diskette, etc.-

En la memoria principal solo pueden guardarse valores mediante su almacenamiento en variables. Por eso, cualquier operación de entrada lleva implícita la asignación del valor introducido en una variable de memoria a la que se deberá hacer referencia cuando se necesite.



4.3.2 Instrucciones

Instrucciones de Comienzo y Fin
Instrucciones de transferencia
Instrucciones de entrada
Instrucciones de salida
Instrucciones de control

Ejemplo
Si se tiene una instrucción de este tipo:
1. Leer (A, B, C)
Lo que se lee es 10, 20, 30 y se asignarán a las variables los siguientes valores:

A = 10
B = 20
C = 30

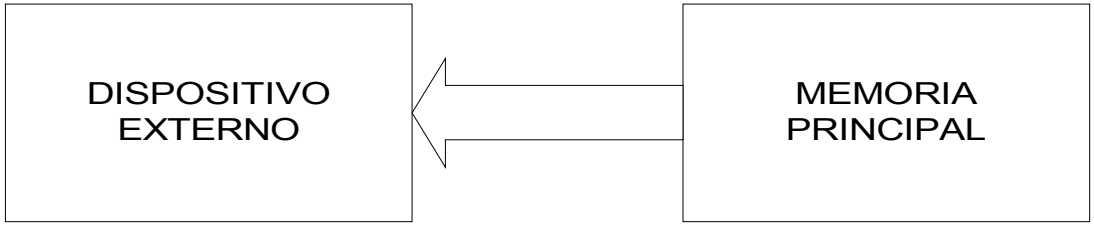
2. Leer (Nombre, Domicilio)
Lo que se lee es Juana, San Juan 1220 y se asignarán a las variables lo siguiente:

Nombre = Juana
Domicilio = San Juan 1220

4.3.2 Instrucciones

Instrucciones de Comienzo y Fin
Instrucciones de transferencia
Instrucciones de entrada
Instrucciones de salida
Instrucciones de control

Permiten la salida de datos desde la memoria principal del ordenador hacia dispositivos externos de salida; por ejemplo impresoras, pantalla, disquete, disco duro, etc



Ejemplo
Si queremos imprimir o visualizar en pantalla los valores de las variables anteriores A, B y C, el resultado sería: 10, 20, 30

4.3.2 Instrucciones

Instrucciones de Comienzo y Fin
Instrucciones de transferencia
Instrucciones de entrada
Instrucciones de salida
Instrucciones de control

El modo en que un ordenador ejecuta las instrucciones contenidas en un programa es, normalmente, secuencial; es decir, una detrás de otra en el orden que están escritas. Sin embargo, si esta fuera la única forma de ejecución posible, el programa tendría que realizar siempre las mismas acciones, independientemente de los datos que se le dieran de entrada en cada ejecución.

Con el fin de poder dotar a los programas de cierta capacidad de decisión sobre los tratamientos que debe aplicar a cada caso, los lenguajes de programación permiten la definición de instrucciones de control distintas a la secuencial. Este es el caso de las instrucciones condicionales e incondicionales.

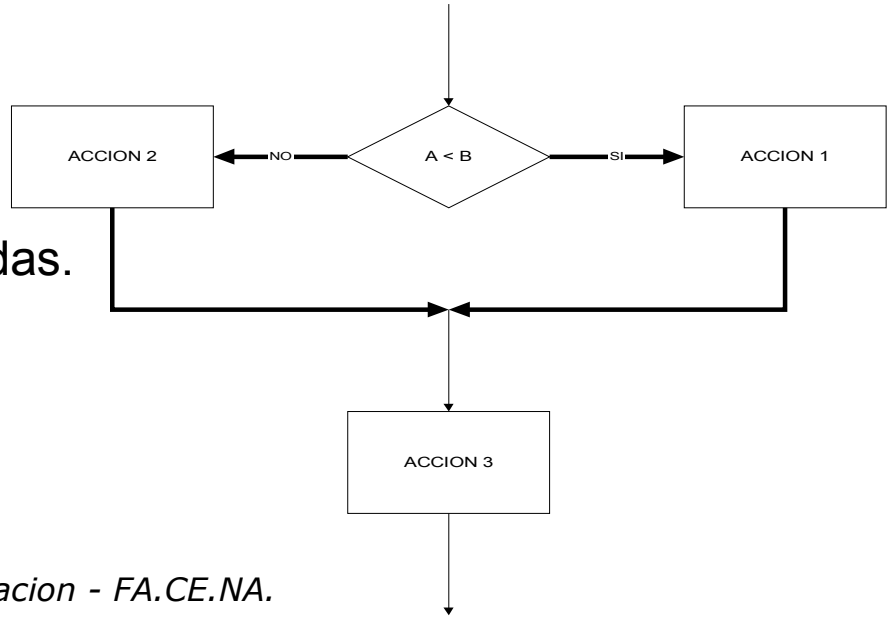
4.3.2 Instrucciones

Instrucciones de Comienzo y Fin
Instrucciones de transferencia
Instrucciones de entrada
Instrucciones de salida
Instrucciones de control

Condicional

Recibe también el nombre de bifurcación condicional, y es aquella que, bajo la comprobación de veracidad o falsedad de una condición, ejecuta dos grupos de acciones diferentes.

En diagramación las interrogaciones se realizan mediante el símbolo de decisión, dentro de este se especifica la comparación deseada con dos salidas.

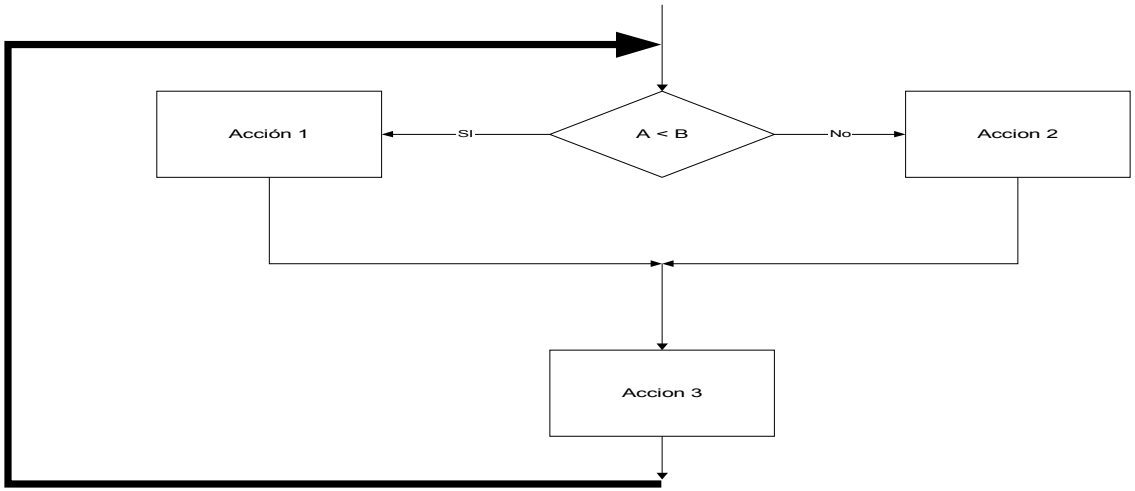


4.3.2 Instrucciones

Instrucciones de Comienzo y Fin
Instrucciones de transferencia
Instrucciones de entrada
Instrucciones de salida
Instrucciones de control

Incondicional

Recibe también el nombre de bifurcación incondicional, e indica un cambio en la secuencia de una ejecución sin evaluar ninguna condición previa.



4.3.3 Elementos básicos de programación

Contadores

Como se mencionó anteriormente, una variable es un campo capaz de almacenar un valor. Este valor, dependiendo de las necesidades del programa, puede variar a lo largo del mismo, por ejemplo muchas veces en los procesos de bucles necesitamos saber el número de iteraciones a realizar por el bucle, o se desea saber cuantos registros hay en un archivo. Una forma de obtener estos resultados es usando un *contador.*

Un contador es una variable numérica cuyo valor se incrementa o decrementa con cantidad fija o constante 1.

La forma de incrementar/decrementar el contador es mediante una instrucción del tipo

$$C = C + 1$$

Siendo C la variable contador, y 1 el incremento/decremento constante.

Los contadores que se utilizan en un programa deben inicializarse con un valor, generalmente cero, que se les asigna fuera del ámbito de la iteración para limpiar la variable de posibles valores anteriores. Además, hay lenguajes que no permiten una instrucción de este tipo si el contador no se encuentra inicializado.

4.3.3 Elementos básicos de programación

Contadores

La acción de inicializar un contador puede ser: C = 0 , CONTA = 0, A = 1

Como los contadores corresponden a las instrucciones del tipo aritméticas, entonces si tenemos la expresión C = C + 1 resulta:

Añadimos el valor 1 al contenido actual de C

Dejamos el nuevo valor otra vez en C

En general, la función más usual de un contador es la de controlar el número de iteraciones que se van realizando en un bucle y asimismo determinar cuando salir de él. También puede irse incrementando o decrementando dentro del bucle cada vez que este se realiza, pero no intervenir en la condición para salir del mismo.

Lo que sí debe quedar claro, es que un contador está siempre asociado a un bucle.

4.3.3 Elementos básicos de programación

Acumuladores

Un acumulador es una variable cuyo valor sé incrementa/decrementa con cantidades variables.

Realmente, realiza la misma función que un contador con la diferencia de que en un acumulador el valor se incrementa o decrementa no es fijo, mientras que en un contador si lo es.

Los acumuladores se usan para calcular totales, entendiendo como total la suma acumulada de diversas cantidades.
La instrucción del acumulador sería:

$$ACUM = ACUM + V$$

Donde ACUM es el acumulador y V el valor variable

Los acumuladores tienen el mismo tratamiento que los contadores en los programas, se deben inicializar en cero fuera del ámbito de la iteración.

4.3.3 Elementos básicos de programación

Acumuladores

Ejemplo: **Se quiere sumar las notas obtenidas por un alumno y las mismas son: 7, 6.50, 8, 9.50.**

Para ello necesitamos una variable numérica que la llamamos SUMANOTAS, que irá conteniendo las sumas parciales que se van obteniendo.
A la variable SUMANOTAS se la supone con un valor inicial 0. Los diferentes valores que irá tomando se muestran en la gráfica siguiente: