Módulo Algoritmos 1

Edgar Eusebio López Murillo
John Jairo Monsalve Restrepo
Luis Fernando González Alvarán
Leonel Osorno Restrepo

Politécnico Colombiano
Jaime Isaza Cadavid

Alcaldía de Medellín
Secretaria de Educación

CONVENIO INTERADMINISTRATIVO
SECRETARÍA DE EDUCACIÓN - MUNICIPIO DE MEDELLÍN
POLITÉCNICO COLOMBIANO JAIME ISAZA CADAVID

Sergio Fajardo Valderrama
Alcalde de Medellín

Juan Camilo Ruíz Pérez
Rector Politécnico Colombiano J.I.C.

Horacio Arango Marín
Secretario de Educación

Giovanni Orozco Arbeláez
Vicerrector de Docencia e Investigación

Ana Lucía Hincapié Correa
Subsecretaria de Planeación Educativa

Gilberto Giraldo Buitrago
Vicerrector de Extensión

Juan Manuel Valdés Barcha Subsecretario de Educación

Darío Castrillón Toro
Director de Proyectos Especiales

Teresita Aguilar García
Subsecretaria Administrativa

Luis Alfredo Aguilar Roldan
Decano Facultad de Ingenierías

Esta es una producción del
Convenio Interadministrativo
Secretaría de Educación Municipio de Medellín
Politécnico Colombiano Jaime Isaza Cadavid.

Coordinadora del convenio
Maryem Aliria Ruiz Nuñez

Autores

Edgar Eusebio López
John Jairo Monsalve Restrepo
Luis Fernando González Alvarán
Leonel Osorno Restrepo
Revisión Luis Fernando González Alvarán

Revisión

Luis Fernando González Alvarán

Impresión
POLITÉCNICO COLOMBIANO JAIME ISAZA CADAVID

Medellín. Febrero de 2007

PLAN DE TRABAJO

INTRODUCCIÓN

El módulo de Algoritmos 1 comprende 6 unidades de trabajo teórico prácticas, que corresponden al contenido curricular de la asignatura Fundamentos de Programación 1 que se orienta en el Politécnico Colombiano Jaime Isaza Cadavid. A través de éste se pretende iniciar al estudiante en el fascinante mundo de la programación, brindándole de una forma amena y sencilla las herramientas básicas y necesarias para que plantee soluciones a partir del uso del computador.

Como afirma "Computer Science" en un informe presentado el 15 de diciembre de 2001, la formación en informática debe presentar especial importancia en temas como: Algoritmos y estructuras de datos, La World Wide Web, tecnología de redes, Bases de datos; entre otros. Es por esto que la media técnica en informática dedica un año completo para el estudio y asimilación de dicha temática, buscando que el estudiante tenga una buena fundamentación que le permita mejorar su nivel de competitividad y eficiencia a la hora de enfrentarse al mercado laboral.

OBJETIVO GENERAL

Enseñar a los estudiantes las estructuras básicas de programación, a partir de conceptos básicos aplicados en un lenguaje algorítmico.

OBJETIVOS ESPECÍFICOS

1. Conocer la forma en que se ingresa, se almacena, se procesa y se muestra la información en una computadora y los elementos que cumplen estas funciones.

2. Evaluar expresiones de cualquier tipo de acuerdo a la prioridad de los operadores y definir los pasos para la solución de un problema cualquiera por medio del computador.

3. Comprender como se realiza la resolución de problemas identificando etapas, estrategias y herramientas lógico matemáticas para aplicarlas a la programación de computadores.

4. Aprender a emplear la estructura secuencial, haciendo uso correcto de las instrucciones de lectura, asignación y escritura.

5. Aprender a emplear la estructura condicional simple y anidada.

6. Aplicar cada una de las estructuras cíclicas en la elaboración de un algoritmo y su proceso de implementación.

INTEGRACIÓN DEL MÓDULO POR UNIDADES

- **UNIDAD 1. Conceptos Básicos De Computación**

  - Organización física de un computador
  - Concepto de lenguaje
  - Sistemas De Numeración
  - Operaciones con Binarios
  - Conversión de un número de una base a otra

- **UNIDAD 2. CONCEPTOS BÁSICOS SOBRE ALGORITMIA**

  - Tipos de datos.
  - Expresiones.
  - Operadores y operandos.
  - Identificadores.
  - Tipos de campos.
  - Definición de Algoritmo.
  - Formas de representación de un algoritmo

- **UNIDAD 3. METODOLOGÍA PARA LA SOLUCIÓN DE PROBLEMAS POR MEDIO DEL COMPUTADOR**

  - Introducción a la solución de problemas.
  - Pasos para la solución de problemas por medio del Computador.

- **UNIDAD 4. ESTRUCTURA SECUENCIAL**

  - Concepto de entrada, asignación y salida de datos.
  - Representación de la estructura secuencial
  - Representación por medio de un Diagrama de Flujo Libre.
  - Representación por medio de un Diagrama de Flujo Estructurado.
  - Representación por medio de Seudocódigo.
  - Ejemplo 1.
  - Ejemplo 2.
  - Ejemplo 3.
  - Ejemplo 4.
  - Ejemplo 5.

INTEGRACIÓN DEL MÓDULO POR UNIDADES

- **UNIDAD 5. ESTRUCTURA CONDICIONAL**

  - Representación Estructura Condicional Simple (Si – entonces).
    Representación por Medio de un Diagrama de Flujo Libre.
    Representación por Medio de un Diagrama de Flujo Estructurado.
    Representación por Medio de Seudocodigo.

  - Representación Estructura Condicional Doble (Si – entonces -sino).
    Representación por Medio de un Diagrama de Flujo Libre.
    Representación por Medio de un Diagrama de Flujo Estructurado.
    Representación por Medio de Seudocodigo.

  - Condicional Múltiple Anidado.
    Representación por Medio de un Diagrama de Flujo Libre.
    Representación por Medio de un Diagrama de Flujo Estructurado.
    Representación por Medio de Seudocodigo.

  - Estructura Según Caso.
    Representación por Medio de un Diagrama de Flujo Libre.
    Representación por Medio de un Diagrama de Flujo Estructurado.
    Representación por Medio de Seudocodigo.

  - Ejemplo 1.
  - Ejemplo 2.
  - Ejemplo 3.
  - Ejemplo 4.
  - Ejemplo 5.
  - Ejemplo 6.

- **UNIDAD 6. ESTRUCTURAS CÍCLICAS O REPETITIVAS**

  - Estructuras cíclicas.
  - Esquemas: cuantitativo y cualitativo.
  - Estructura Mientras.
  - Rompimientos de: ciclos y control de ejecución.
  - Estructuras Para
  - Estructura Hacer mientras que.

- **BIBLIOGRAFÍA**

UNIDAD 1

CONCEPTOS BÁSICOS DE COMPUTACIÓN

INTRODUCCIÓN

Una Computadora es una máquina que permite: (1) realizar procesos para darnos resultados sin que veamos las operaciones que realiza, (b) diseñar soluciones a la medida de problemas específicos que se nos presenten, y (c) manejar un volumen muy grande de datos.

JUSTIFICACIÓN

El diseño de soluciones a la medida de nuestros problemas, requiere una metodología que nos enseñe de manera gradual, la forma de llegar a estas soluciones.

A las soluciones creadas por computadora se les conoce como **programas** y no son más que una serie de operaciones que realiza la computadora para llegar a un resultado (o solución), con un grupo de datos específicos.

Para poder realizar programas, además de conocer la metodología, también debemos conocer, de manera específica las funciones que realiza la computadora, la forma de almacenar la información en la memoria de la misma y las formas en que se pueden manejar los elementos que hay en la misma.

OBJETIVO GENERAL

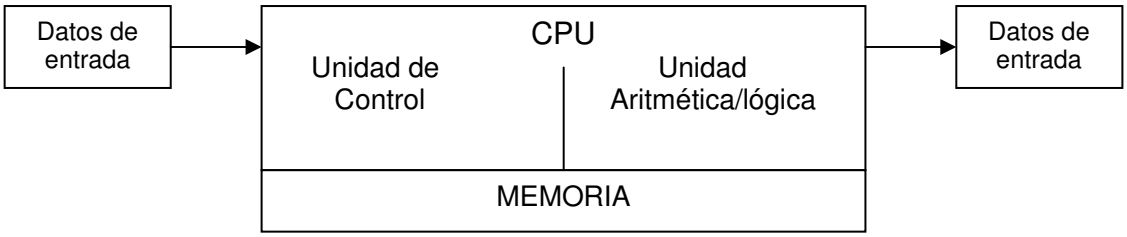
Conocer la forma en que se ingresa, se almacena, se procesa y se muestra la información en una computadora y los elementos que cumplen estas funciones.

OBJETIVOS ESPECÍFICOS

1. Identificar las partes básicas en que se divide el computador.
2. Conocer los diferentes sistemas de numeración y las conversiones de una base a otra.
3. Conocer la terminología relacionada con los algoritmos; así como la importancia de aplicar técnicas adecuadas de programación.
4. Conocer la metodología para resolver problemas por medio del computador en cada una de sus etapas.

CONTENIDO

1. ORGANIZACIÓN FÍSICA DE UN COMPUTADOR



Dispositivos de Entrada: Sirven para introducir datos (información) en la computadora para su proceso. Los datos se leen de los dispositivos de entrada y se almacenan en la memoria central o interna. Ejemplos: teclado, scanners lápiz digitalizador, mouse (ratón), trackball (bola de ratón estacionario), joystick (palancas de juego), lápiz óptico, etc.

Dispositivos de Salida: Regresan los datos procesados que sirven de información al usuario. Ejemplo: monitor, impresora.

La Unidad Central de Procesamiento (C.P.U) se divide en dos:
- Unidad de control
- Unidad Aritmética - Lógica

Unidad de Control: Coordina las actividades de la computadora y determina que operaciones se deben realizar y en que orden; así mismo controla todo el proceso de la computadora.

Unidad Aritmética - Lógica: Realiza operaciones aritméticas y lógicas, tales como suma, resta, multiplicación, división y comparaciones.

La Memoria de la computadora se divide en dos:
- Memoria Central o Interna
- Memoria Auxiliar o Externa

Memoria Central (interna): La CPU utiliza la memoria de la computadora para guardar información mientras trabaja con ella; mientras esta información permanezca en memoria, la computadora puede tener acceso a ella en forma directa. Esta memoria construida internamente se llama memoria de acceso aleatorio (RAM). La memoria interna consta de dos áreas de memoria:

La memoria RAM (Randon Access Memory): o memoria principal. En ella se almacena información solo mientras la computadora está encendida. Cuando se apaga, la información se pierde, por lo que se dice que la memoria RAM es una memoria volátil.

La memoria ROM (Read Only Memory): Es una memoria estática que no puede cambiar, la computadora puede leer los datos almacenados en la memoria ROM, pero no se pueden introducir datos en ella, o cambiar los datos que ahí se encuentran; por lo que se dice que esta memoria es de solo lectura. Los datos de la memoria ROM están grabados en forma permanente y son introducidos por el fabricante de la computadora.

Memoria Auxiliar (Externa): Es donde se almacenan todos los programas o datos que el usuario desee. Los dispositivos de almacenamiento o memorias auxiliares (externas o secundarias) más comúnmente utilizados son: cintas magnéticas y discos magnéticos, discos ópticos y memorias USB (también llamados memory key).

2. CONCEPTO DE LENGUAJES

Lenguaje: Es una serie de símbolos que sirven para transmitir uno o mas mensajes (ideas) entre dos entidades diferentes. A la transmisión de mensajes se le conoce comúnmente como comunicación.

La comunicación es un proceso complejo que requiere una serie de reglas simples, pero indispensables para poderse llevar a cabo. Las dos principales son las siguientes:

* Los mensajes deben correr en un sentido a la vez.
* Debe forzosamente existir 4 elementos: Emisor, Receptor, Medio de Comunicación y Mensaje.

Lenguajes de Programación: Es un conjunto de símbolos, caracteres y reglas (o sintaxis) que le permiten a las personas comunicarse con la computadora.

Los lenguajes de programación tienen un conjunto de instrucciones que nos permiten realizar operaciones de entrada/salida, cálculo, manipulación de textos, lógica/comparación y almacenamiento/recuperación.

Los lenguajes de programación se clasifican en:

- **Lenguaje Máquina:** Son aquellos cuyas instrucciones son directamente entendibles por la computadora y no necesitan traducción posterior para que la CPU pueda

comprender y ejecutar el programa. Las instrucciones en lenguaje maquina se expresan en términos de la unidad de memoria mas pequeña el bit (dígito binario 0 o 1). Este lenguaje, denominado binario es el único que entiende el procesador del computador, ya que la memoria de este es un grupo de elementos biestables (dos estados) que son 0 (apagado) y 1 (prendido), o sea, lo que se guarda son ceros y unos.

- **Lenguaje de Nivel intermedio** (Ensamblador): En este lenguaje las instrucciones se escriben en códigos alfabéticos conocidos como mnemotécnicos o mnemónicos para las operaciones y direcciones simbólicas.

- **Lenguaje de Alto Nivel**: Los lenguajes de programación de alto nivel (BASIC, pascal, cobol, fortran, C++, Java, etc.) son aquellos en los que las instrucciones o sentencias a la computadora son escritas con palabras similares a los lenguajes humanos (en general en inglés, aunque ya hay en español), lo que facilita la escritura y comprensión del programa.

En síntesis se pude decir que la comunicación usuario computador se da a partir de un lenguaje natural que está del lado del usuario, el cual por intermedio de herramientas lógico matemáticas se convierte en unos y ceros para estar del lado de la maquina, como se puede observar en el siguiente grafico:

Unidades de información

Cada trozo de información recibe un nombre propio según la cantidad de bits que posea:

Un **bit** es la unidad de información binaria y con él se puede contar desde 0 hasta 1.

Un **nibble** son cuatro bits y se puede contar desde 0 hasta 15 (0xF en hexa).

Con un **byte** (8 bits) se puede contar desde 0 hasta 255 ó 0xFF hexa.

Una **word** tiene 16 bits y permite contar desde 0 hasta 65535 ó 0xFFFF.

Una **double-word** (32 bits) permite contar desde 0 hasta 4.294.967.295 ó 0xFFFFFFFF.

Cuando usted escuche hablar de direcciones de 32 bits, sepa que hay un espacio de almacenamiento de 4.294 ... millones de bytes o 4 Gigabytes (o de colores, si estamos hablando de color de 32 bits).

3. SISTEMAS DE NUMERACIÓN

Un sistema numérico en una base cualquiera ***b*** es un sistema que utiliza distintos símbolos para representar sus ***b*** dígitos. Los números se representan mediante cadenas de símbolos de dígitos. Para determinar la cantidad que el número representa es necesario multiplicar cada dígito por una potencia entera de b y luego efectuar la suma de todos los dígitos pesados. Un dígito pesado es aquel que tiene dos valores: unos intrínseco y otro posicional).

El sistema numérico decimal, que es el sistema usado por nosotros a diario en nuestras operaciones aritméticas, está en base 10 (b=10), lo cual significa que utiliza 10 (0, 1, 2, 3, 4, 5, 6, 7, 8, 9) dígitos para representar cualquier cantidad posible. Así, la cadena de números 605.3 representa la cantidad:

$$6 * 10^2 + 0 * 10^1 + 5 * 10^0 + 3 * 10^{-1}$$

o sea, 6 centenas más 0 decenas más 5 unidades más 3 décimas.

Sistema binario:

Es el sistema utilizado por el computador y llamado también lenguaje de máquina. Utiliza dos dígitos (b = 2) que son el 0 y el 1. La cadena de números 1101101 en base 2, que se representa también como $(1101101)_2$, representa la cantidad:

$1 * 2^6 + 1 * 2^5 + 0 * 2^4 + 1 * 2^3 + 1 * 2^2 + 0 * 2^1 + 1 * 2^0 = 64 + 32 + 0 + 8 + 4 + 0 + 1 = 109$

O sea, $(1101101)_2 = (109)_{10}$

Sistema octal:

Algunos procesadores forman grupos de tres dígitos binarios para formar un dígito octal ($2^3 = 8^1$), para trabajar con cadenas de una longitud de la tercera parte de la que tendría si trabajara con cadenas de 0's y 1's
El sistema octal consta de ocho dígitos que son 0, 1, 2, 3, 4, 5, 6, 7 y que corresponden en binario a las siguientes cadenas de 0's y 1's:

$(0)_8 = (000)_2$	$(1)_8 = (001)_2$
$(2)_8 = (010)_2$	$(3)_8 = (011)_2$
$(4)_8 = (100)_2$	$(5)_8 = (101)_2$
$(6)_8 = (110)_2$	$(7)_8 = (111)_2$

Sistema hexadecimal:

Otros procesadores forman grupos de cuatro dígitos binarios para formar un dígito hexadecimal ($2^4 = 16$), para trabajar con cadenas de una longitud de la cuarta parte de la que tendría si trabajara con cadenas de 0's y 1's.

El sistema hexadecimal consta de diez y seis dígitos que son 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F y que corresponden en binario a las siguientes cadenas de 0's y 1's:

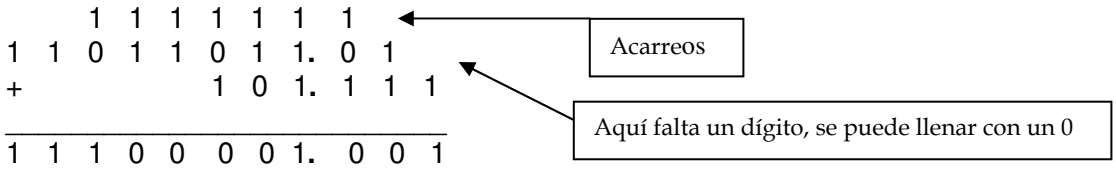
$(0)_{16} = (0000)_2$	$(1)_{16} = (0010)_2$	$(2)_{16} = (0010)_2$	$(3)_{16} = (0011)_2$
$(4)_{16} = (0100)_2$	$(5)_{16} = (0101)_2$	$(6)_{16} = (0110)_2$	$(7)_{16} = (0111)_2$
$(8)_{16} = (1000)_2$	$(9)_{16} = (1001)_2$	$(A)_{16} = (1010)_2$	$(B)_{16} = (1011)_2$
$(C)_{16} = (1100)_2$	$(D)_{16} = (1101)_2$	$(E)_{16} = (1110)_2$	$(F)_{16} = (1111)_2$

4. Operaciones Con Binarios

Suma de binarios

Para sumar números binarios hay que tener en cuenta las siguientes reglas:
0 + 0 = 0
0 + 1 = 1
1 + 0 = 1
1 + 1 = 10 (se lee 0 y 1 de acarreo)
Es decir, cuando se suman dos unos, el resultado da 0 y se lleva 1 de acarreo. Así mismo, cuando se suman tres unos el resultado es 1 y se lleva uno de acarreo. Cuando se suman cuatro unos el resultado es 0 y se llevan dos acarreos de a uno y así sucesivamente. Hay que tener en cuenta que para sumar o restar números binarios, se debe alinear en una misma columna el punto decimal.

Ejemplo: sumar los números binarios 11011011.01 y 101.111



Resta de binarios

Para restar números binarios hay que tener en cuenta las siguientes reglas:

0 - 0 = 0
0 - 1 = No se puede.
1 - 0 = 1
1 - 1 = 0

Ejemplo: Restar los números binarios 1001001 (73 en decimal) y 10101 (21 en decimal)

Para restar números binarios es necesario primero expresar el sustraendo (o los dos números si son negativos) a complemento a uno o a complemento a dos. Veamos primero como se convierte un número binario a complemento a uno.

Para convertir un número a complemento a uno, se resta cada dígito del número 1, así por ejemplo, el complemento a uno de 10101 con 7 bits es: (siempre hay que completar los bits con ceros)

```
1   1   1   1   1   1   1
0   0   1   0   1   0   1
_____________________________
1   1   0   1   0   1   0   Complemento a uno del binario 10101 con 7 bits
```

Claramente se ve que para hallar el complemento a uno de un número binario, se cambian los unos por ceros y los ceros por unos.
Para hallar el complemento a dos de un número binario, se le suma 1 al complemento a uno. Así, el complemento a dos del binario 10101 con 7 bits es:

```
1   1   0   1   0   1   0
                    1
_____________________________
1   1   0   1   0   1   1   Complemento a dos del binario 10101 con 7 bits
```

Una vez hallado el complemento a uno o el complemento a dos del sustraendo, se puede hacer la resta de binarios así:

a. Si se utiliza el complemento a uno, se suma el minuendo con el complemento a uno del sustraendo. Si al final da un acarreo, se suma a la suma anterior y el resultado será la diferencia. Siguiendo con el ejemplo anterior (1001001 – 10101):

```
1   0   0   1   0   0   1   (73 en decimal)
1   1   0   1   0   1   0   (C'1 del sustaendo con 7 bits)
_____________________________
0   1   1   0   0   1   1
                        1   (Acarreo)
_____________________________
0   1   1   0   1   0   0   (52 en decimal)
```

b. Si se utiliza el complemento a dos, se suma el minuendo con el complemento a dos del sustraendo. Si al final da un acarreo, se desprecia, es decir, no se tiene en cuenta. Siguiendo con el ejemplo anterior:

```
1   0   0   1   0   0   1   (73 en decimal)
1   1   0   1   0   1   1   C'2 del binario 10101 con 7 bits
_____________________________
0   1   1   0   0   1
```

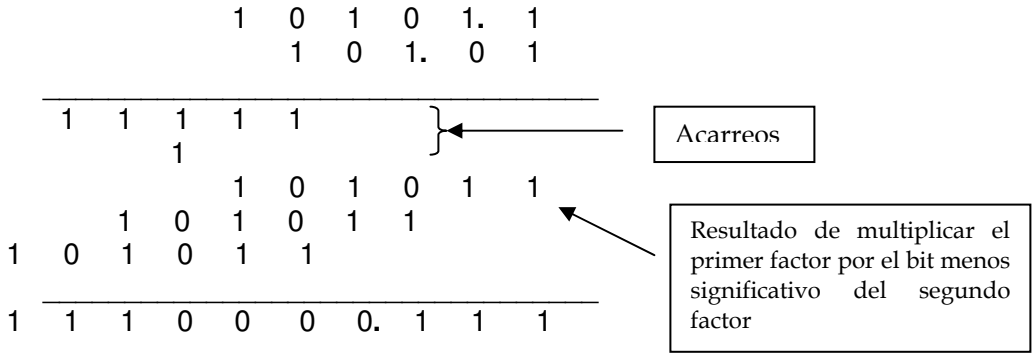
Sumando los últimos bits 1+1 dio un acarreo que no se tiene en cuenta.

Multiplicación de binarios

Para multiplicar números binarios hay que tener en cuenta las siguientes reglas:

0 * 0 = 0
0 * 1 = 0
1 * 0 = 0
1 * 1 = 1

Ejemplo: Multiplicar los números binarios 10101.1 y 101.01



Observe que, igual que hacemos en la multiplicación en base 10, se suman los dígitos decimales de cada factor y ese número de decimales se separan en el resultado.

División de binarios

Ya que la división no es más que multiplicaciones y restas, para dividir números binarios no existen reglas propias, se rige por las normas de la multiplicación y de la resta.

Hay que tener presente que el divisor debe ser un número entero, sin parte decimal, si no lo es, se convierte trasladando el punto decimal el número de lugares a la derecha como decimales tenga y la misma cantidad de lugares debe trasladarse en el dividendo.

Para hacer la división:

a. se cuentan los dígitos del divisor y la misma cantidad de dígitos se toma en el dividendo.

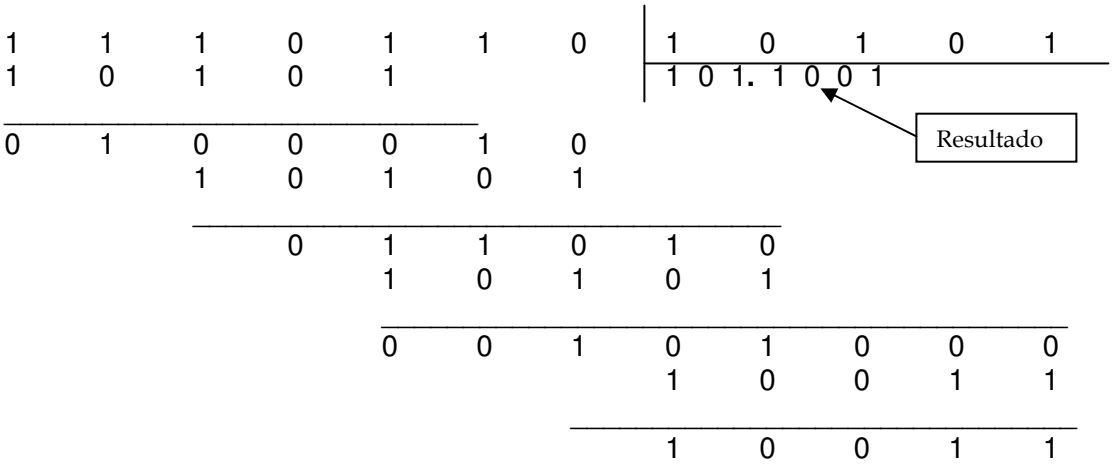
b. se comparan dígito a dígito para estar seguros de que el grupo de bits separados en el dividendo es mayor o igual al grupo de bits que hay en el divisor

c. si lo es, se coloca 1 en el cociente, si no lo es, se toma un dígito más en el dividendo, con lo cual estamos seguros que el dividendo cabe una vez en el grupo tomado en el divisor.

d. luego se multiplica el 1 que hay en el cociente por el divisor

e. se coloca el resultado debajo del grupo tomado en el dividendo y se hace la resta
f. el resultado se coloca debajo y se aumenta (se baja) con el siguiente dígito del dividendo y
g. se repiten los pasos *a* al *f*
h. si se encuentra el punto decimal se coloca en el cociente
i. si se requiere sacar cifras decimales se agrega un cero al grupo de bits del divisor y se procede igual que en los pasos *a* al *f*

Ejemplo: Dividir los números binarios 11101.1 y 101.01



Hay cinco operaciones binarias básicas: AND, OR, NOT, XOR y ADD. La resta, multiplicación y división se derivan de estas cinco anteriores. Cualquiera sea la longitud de la palabra o palabras objeto de la operación, siempre se hace de a un bit por vez de derecha a izquierda (tal como si fuera una suma o resta con números decimales). Esto permite una definición de cada operación que es independiente de la longitud del o de los operando(s). La operación NOT es la única que se realiza sobre un sólo operando (es unaria), y las otras cuatro sobre dos operandos.

La operación AND (Y) tiene resultado 1 si sus dos operandos son ambos 1
La operación OR (O) tiene resultado 1 si cualquiera de sus operandos es 1
La operación XOR tiene resultado 1 si los operandos son distintos (uno en 0 y el otro en 1).

La operación NOT (NO) tiene resultado 1 si el operando es 0 y viceversa
La operación ADD (SUMA) se define igual que con los números decimales

AND	OR	XOR	NOT	SUMA
0 * 0 = 0	0 + 0 = 0	0 X 0 = 0	NOT 1 = 0	0 + 0 = 0
0 * 1 = 0	0 + 1 = 1	0 X 1 = 1	NOT 0 = 1	0 + 1 = 1
1 * 0 = 0	1 + 0 = 1	1 X 0 = 1	---	1 + 0 = 1
1 * 1 = 1	1 + 1 = 1	1 X 1 = 0	---	1 + 1 = 10

5. CONVERSIÓN DE UN NÚMERO DE UNA BASE A OTRA

✓ *Conversión de binario a decimal*

Se puede convertir un número binario (base 2) en número decimal (base 10), multiplicando cada dígito binario por una potencia de 2 que corresponde a su posición dentro del número, así: desde el punto decimal hacia la izquierda las potencias de 2 son 2^0 2^1 2^2 2^3 2^4 2^5 y así sucesivamente y desde el punto decimal a hacia la derecha son 2^{-1} 2^{-2} 2^{-3} 2^{-4} y así sucesivamente. Por ejemplo para convertir el número binario 110101.01 en número decimal se procede así:

2^5	2^4	2^3	2^2	2^1	2^0	2^{-1}	2^{-2}	Potencias de 2
↓	↓	↓	↓	↓	↓	↓	↓	
1	1	0	1	0	1 .	0	1	Dígitos binarios
↓	↓	↓	↓	↓	↓	↓	↓	
1*32	+1*16	+0*8	+ 1*4	+ 0*2	+ 1*1	+ 0*0.5	+	1*0.25
↓	↓	↓	↓	↓	↓	↓	↓	
32	+ 16	+ 0	+ 4	+ 0	+ 1	+ 0	+ 0.25	= $(53.25)_{10}$

✓ *Conversión de decimal a binario*

Para convertir un número decimal a número binario se procede en una forma para la parte entera y en forma diferente para la parte fraccionaria, así:
La parte entera se divide sucesivamente por 2 hasta que el último cociente sea 0 y el binario resultante será conformado por el último residuo, luego el penúltimo residuo y así sucesivamente hasta tomar el primer residuo.
La parte fraccionaria se multiplica por 2 y se toma el número entero resultante para formar parte del número binario resultante, luego se multiplica nuevamente la parte fraccionaria por 2 y se toma la parte entera para formar parte del binario resultante, y así sucesivamente hasta que se presente alguno de los siguientes casos: a) la parte fraccionaria es 0.0, b) se descubre que se repite la fracción, en cuyo caso se reproduce un período, c) se dice de antemano cuántas cifras decimales se consiguen.

Por ejemplo si se quiere convertir el número decimal 57.25 en número binario se procede así:

La parte entera del número decimal (base 10):

57 / 2 = 28 (residuo = 1)
28 / 2 = 14 (residuo = 0)
14 / 2 = 7 (residuo = 0)
7 / 2 = 3 (residuo = 1)
3 / 2 = 1 (residuo = 1)
1 / 2 = 0 (residuo = 1)

En este sentido se toman los dígitos

La parte fraccionaria del número decimal:

0.25 * 2 = 0.50 (la parte entera del resultado es 0)
0.5 * 2 = 1.0 (la parte entera del resultado es 1) ↓ En este sentido se toman los dígitos

Luego, el binario resultante es 111001.01

- ✓ ***Conversión de octal a decimal***

Se convierte un número octal (base 8) en número decimal (base 10), multiplicando cada dígito binario por una potencia de 8 que corresponde a su posición dentro del número, así, desde al punto decimal de derecha a izquierda las potencias de 8 son 8^0 8^1 8^2 8^3 8^4 8^5 y desde el punto decimal de izquierda a derecha son 8^{-1} 8^{-2} 8^{-3} 8^{-4} y así sucesivamente.

Por ejemplo para convertir el número octal 175.04 a número decimal se procede así:

8^2		8^1		8^0		8^{-1}		8^{-2}	→	Potencias de 8	
↓		↓		↓		↓		↓			
1		7		5		0		4	→	Dígitos octales	
↓		↓		↓		↓		↓			
1*64	+	7*8	+	5*1	+	0*0.125	+	4*0.015625			
↓		↓		↓		↓		↓			
64	+	56	+	5	+	0	+	0.0625	=	$(125.0625)_{10}$	

- ✓ ***Conversión de decimal a octal***

Para convertir un número decimal a número octal se procede en una forma para la parte entera y en forma diferente para la parte fraccionaria, así:

La parte entera se divide sucesivamente por 8 hasta que el último cociente sea 0 y el octal resultante será conformado por el último residuo, luego el penúltimo residuo y así sucesivamente hasta tomar el primer residuo.

La parte fraccionaria se multiplica por 8 y se toma el número entero resultante para formar parte del número octal resultante, luego se multiplica nuevamente la parte fraccionaria por 8 y se toma la parte entera para formar parte del octal resultante, y así sucesivamente hasta que se presente alguno de los siguientes casos: a) la parte fraccionaria es 0.0, b) se descubre que se repite la fracción, en cuyo caso se reproduce un período, c) se dice de antemano cuántas cifras decimales se consiguen.

Por ejemplo si se quiere convertir el número decimal 139.135 en número binario se procede así:

La parte entera del número decimal (base 10):

139 / 8 = 17 (residuo = 3)
17 / 8 = 2 (residuo = 1) En este sentido se toman los dígitos
2 / 8 = 0 (residuo = 2)

La parte fraccionaria del número decimal:

0.135 * 8 = 1.08 (la parte entera del resultado es 1)
0.08 * 8 = 0.64 (la parte entera del resultado es 0)
0.64 * 8 = 5.12..... (la parte entera del resultado es 5) En este sentido se toman los dígitos
0.12 * 8 = 0.96..... (la parte entera del resultado es 0)
0.96 * 8 = 7.68 (la parte entera del resultado es 7)
0.68 * 8 = 5.44 (la parte entera del resultado es 5)

y así sucesivamente.

Luego, el octal resultante es 213.105075

- ✓ ***Conversión de binario a octal:***

Ya que $2^3 = 8^1$, tres dígitos binarios se pueden representar como un dígito octal Por lo tanto lo que haremos es representar cada grupo de 3 dígitos binarios como un dígito octal así: se separan grupos de tres binarios tomados desde el punto decimal hacia la izquierda y desde el punto decimal hacia la derecha y se convierte cada grupo de tres en un dígito octal multiplicando cada dígito binario por su respectiva potencia de 2 de acuerdo con su posición dentro de ese grupo. Si algún grupo está incompleto, se completa con el o los ceros necesarios.

Ejemplo, convertir el número binario 1100111101.0101 a número octal.

Se conforman los grupos de tres binarios así:

001 100 111 101. 010 100

Observe que el primer uno queda sólo y se completa el grupo de tres agregándole dos ceros a la izquierda, lo que no afecta su valor. Lo mismo ocurre con el último uno de la derecha el cual se completa con dos ceros a la derecha.

Ahora se evalúa cada grupo independientemente de su posición dentro del número peor multiplicando cada dígito binario por su potencia de dos respectiva así, para el primer grupo de tres dígitos sería:

2^2 2^1 2^0
↓ ↓ ↓
0 0 1
↓ v ↓
0*4 0*2 1*1
↓ ↓ ↓
0 + 0 + 1 = 1

Para el segundo grupo de tres dígitos binarios sería:

2^2 2^1 2^0
↓ ↓ ↓
1 0 0
↓ v ↓
1*4 0*2 0*1
↓ ↓ ↓
4 + 0 + 0 = 4

Para el tercer grupo de tres dígitos binarios sería:

2^2 2^1 2^0
↓ ↓ ↓
1 1 1
↓ v ↓
1*4 1*2 1*1
↓ ↓ ↓
4 + 2 + 1 = 7

Para el cuarto grupo de tres dígitos binarios sería:

2^2 2^1 2^0
↓ ↓ ↓
1 0 1
↓ v ↓
1*4 0*2 1*1
↓ ↓ ↓
4 + 0 + 1 = 5

Luego va el punto decimal y para el primer grupo de tres dígitos binarios sería después del punto decimal:

2^2 2^1 2^0
↓ ↓ ↓
0 1 0
↓ v ↓
0*4 1*2 0*1
↓ ↓ ↓
0 + 2 + 0 = 2

Para el segundo grupo de tres dígitos binarios después del punto decimal sería:

2^2 2^1 2^0
↓ ↓ ↓
1 0 0
↓ ↓ ↓
1*4 0*2 0*1
↓ ↓ ↓
4 + 0 + 0 = 4

Luego el binario 1100111101.0101 es equivalente al número octal 1475.24.

- ✓ ***Conversión de octal a binario:***

Aquí procederemos en sentido contrario, es decir convertiremos un dígito octal en tres dígitos binarios. El grupo de tres binarios se colocarán en el mismo sitio donde estaba el octal del cual resultó.

Ejemplo, convertir el número octal 642.05 a binario.

6 4 2**.** 0 5
↓ ↓ ↓ ↓ ↓
110 100 010 000 101

El binario resultante es 110100010**.**000101

Como ejercicio el estudiante consultará la conversión de binario a hexadecimal y de

hexadecimal a binario.

La siguiente tabla muestra la correspondencia entre los diferentes sistemas de numeración:

Número Decimal	Número binario	Número Octal	Número Hexadecimal
0	0	0	0
1	1	1	1
2	10	2	2
3	11	3	3
4	100	4	4
5	101	5	5
6	110	6	6
7	111	7	7
8	1000	10	8

Número Decimal	Número binario	Número Octal	Número Hexadecimal
9	1001	11	9
10	1010	12	A
11	1011	13	B
12	1100	14	C
13	1101	15	D
14	1110	16	E
15	1111	17	F
16	10000	20	10
17	10001	21	11
18	10010	22	12
19	10011	23	13
20	10100	24	14
21	10101	25	15
22	10110	26	16
23	10111	27	17
24	11000	30	18
25	11001	31	19
26	11010	32	1 A
27	11011	33	1B
28	11100	34	1C
29	11101	35	1D
30	11110	36	1E
31	11111	37	1F
32	100000	40	20
33	100001	41	21
34	100010	42	22
35	100011	43	23
36	100100	44	24
37	100101	45	25
38	100110	46	26
39	100111	47	27
40	101000	50	28
41	101001	51	29
42	101010	52	2 A
43	101011	53	2B
44	101100	54	2C
45	101101	55	2D
46	101110	56	2E

ACTIVIDADES

1. Responder al siguiente cuestionario

1. Introducir datos (información) en la computadora para un proceso. lo realiza la

   a. *Unidad de entrada.*
   b. *Unidad de memoria*
   c. *Unidad de almacenamiento*

2. La Coordinación de las actividades de la computadora y la administración de las operaciones y procesos ordenadamente se denomina

   a. *Unidad de entrada*
   b. *Unidad de control*
   c. *Unidad de proceso*

3. Realizar operaciones aritméticas y lógicas *esta encargado a la ¿:*

   a. *Unidad aritmética.*
   b. *Unidad aritmética y de procesos.*
   c. *Unidad aritmética y lógica.*

4. La sigla C P U traduce:

   a. *Unidad de procesamiento centrado y se descompone en 2 partes*
   b. *Unidad central de procesamiento y se descompone en 2 partes*
   c. *Unidad central de procesamiento y descompone en 3 partes*

5. ¿Por cuál número potencia que corresponde a su posición dentro del número se debe multiplicar cada digito binario para convertir un número binario en número decimal?

   a. *Numero 10*
   b. *Numero 2*
   c. *Numero 2 y 10*

2. Consultar los siguientes temas.

   a. *Conversión de hexadecimal a decimal.*
   b. *Conversión de decimal a hexadecimal.*

ESTRATEGIAS METODOLÓGICAS

- ✓ Presentación de la unidad por parte del profesor.
- ✓ Explicación de las funciones de cada parte del computador
- ✓ Identificar las partes del computador y sus funciones, aprendiendo a manipularlos
- ✓ Explicación del concepto de lenguaje, lenguaje de programación, lenguaje de máquina, etcétera.
- ✓ Realización de ejemplos de conversión de un número en una base a otro equivalente en otra base.
- ✓ Realización de ejemplos de las diferentes operaciones entre números binarios.

RECURSOS

- ✓ Humanos: profesor y alumnos.
- ✓ Institucionales: Salón de clases.
- ✓ Materiales: Módulo guía y tablero.

INDICADORES DE EVALUACIÓN

- ✓ Evaluación escrita sobre la unidad

CRITERIOS DE EVALUACIÓN

- ✓ Estar en capacidad de enumerar las partes que componen el hardware del computador.
- ✓ Enumerar las partes que componen el software del computador.
- ✓ Convertir un número binario (base 2) a decimal (base 10).
- ✓ Convertir un número decimal (base 10) a binario (base 2).

- ✓ Convertir un número hexadecimal (base 16) a base 2.
- ✓ Convertir un número binario a hexadecimal.
- ✓ Convertir un número en una base cualquiera a otra diferente.
- ✓ Sumar dos o más números binarios.
- ✓ Restar dos números binarios.
- ✓ Multiplicar dos números binarios.
- ✓ Dividir dos números binarios.

UNIDAD 2

CONCEPTOS BÁSICOS SOBRE ALGORITMIA

INTRODUCCIÓN

Los programas de computador a menudo contienen algunos elementos llamados **campos**, que son espacios de memoria en los cuales el usuario puede guardar datos, ya sea que el los ingrese o que el programa los guarde como resultado de alguna operación. Estos campos pueden ser constantes o variables y pueden almacenar datos de tipo numérico, de tipo carácter o de tipo lógico.

JUSTIFICACIÓN

Para elaborar un programa se debe definir los campos (nombre y tipo), es decir, hay que darle un nombre de identificador y se debe establecer el tipo de dato que se va a almacenar en él. Así mismo, se debe saber como se escriben las expresiones en un algoritmo y conocer las diferentes formas de representar un algoritmo.

OBJETIVO GENERAL

Evaluar expresiones de cualquier tipo de acuerdo a la prioridad de los operadores y definir los pasos para la solución de un problema cualquiera por medio del computador.

OBJETIVOS ESPECÍFICOS

1. Reconocer los tipos de campos que maneja el computador.
2. Identificar los tipos de variables que puede utilizar en un algoritmo.
3. Identificar los operadores y operandos que hay en una expresión.
4. Conocer la prioridad de los operadores.
5. Conocerá las reglas para cambiar formulas matemáticas a expresiones válidas para la computadora
6. Conocer la metodología utilizada en la elaboración de algoritmos.
7. Conocer las diferentes formas de representar un algoritmo.

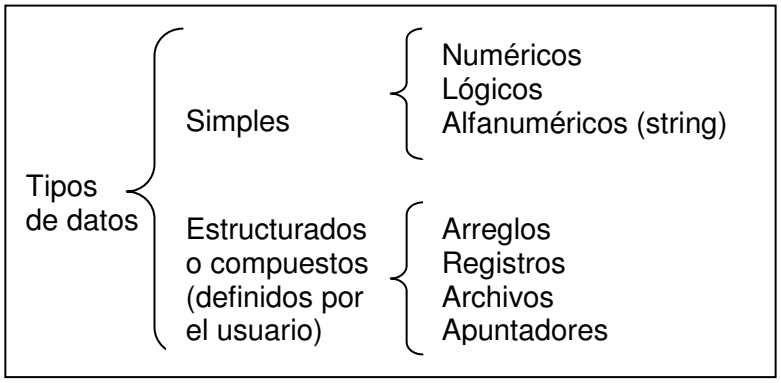
CONTENIDO

1. Tipos de datos.
2. Expresiones.
3. Operadores y operandos.
4. Identificadores.
5. Tipos de campos.
6. Definición de Algoritmo.
7. Formas de representación de un algoritmo.

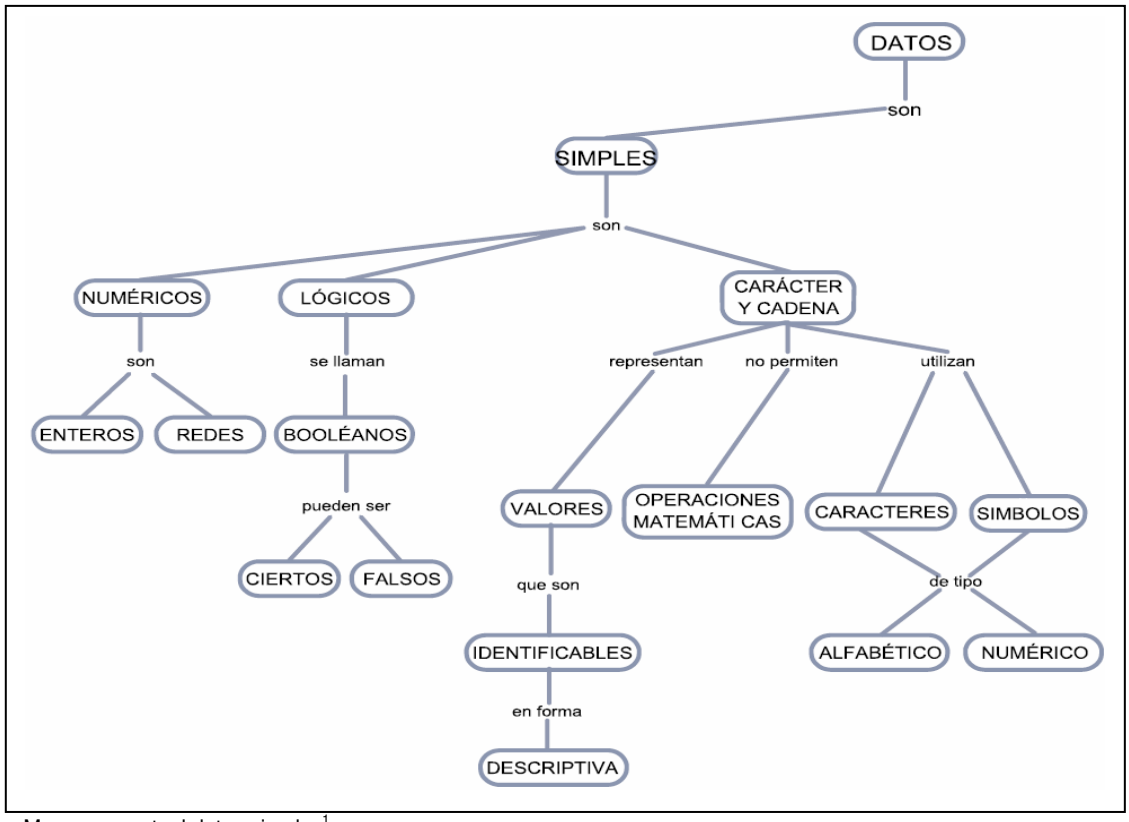
1. TIPOS DE DATOS

Todos los datos tienen un tipo asociado con ellos. Un dato puede ser un simple carácter, tal como 'b', un valor entero tal como 35. El tipo de dato determina la naturaleza del conjunto de valores que puede tomar una variable.

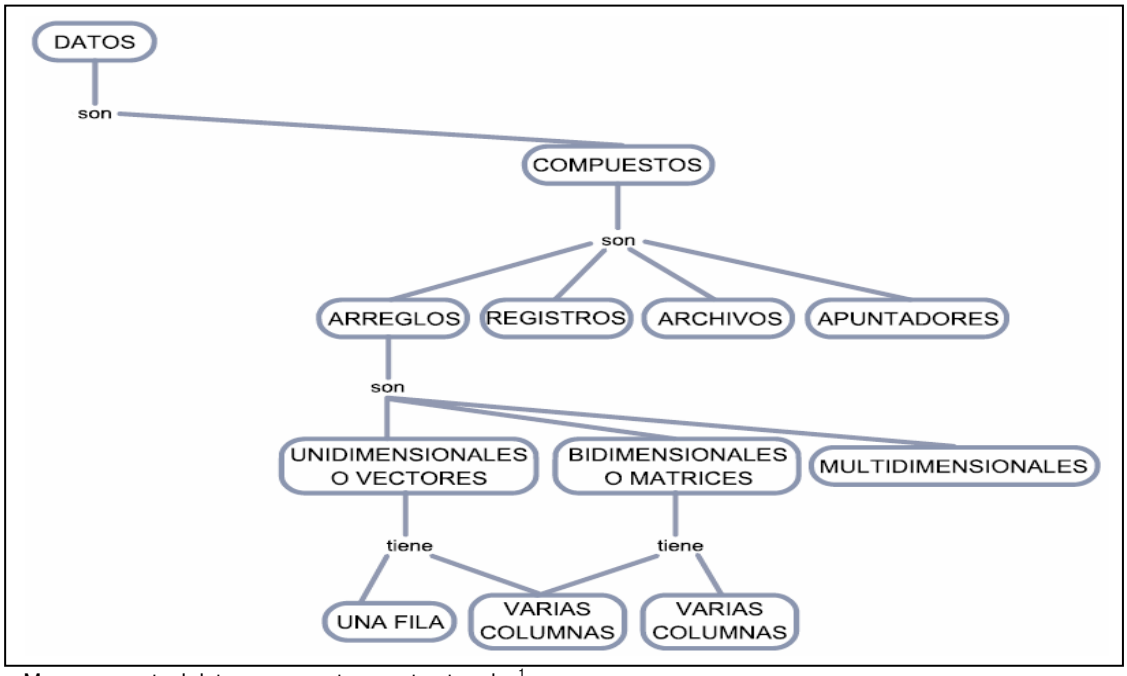
La clasificación de los datos según su tipo es:



Aunque este curso se centra principalmente en los datos simples, se han planteado dos mapas conceptuales (uno para los simples y otro para los compuestos) que puede ayudar al entendimiento de su connotación y clasificación). Estos se pueden observar a continuación



Mapa conceptual datos simples[1]



Mapa conceptual datos compuestos o estructurados[1]

[1] Tomado del libro "Fundamentos de programación con énfasis en análisis y metodología para trabajo en equipos efectivos

➢ **Datos Numéricos**: Permiten representar valores escalares de forma numérica, esto incluye a los números enteros y los reales. Este tipo de datos permiten realizar operaciones aritméticas comunes.

➢ **Datos Lógicos**: Son aquellos que solo pueden tener dos valores (verdadero o falso) ya que representan el resultado de una comparación entre otros datos (numéricos o alfanuméricos).

➢ **Datos Alfanuméricos (String)**: Es una secuencia de caracteres alfanuméricos que permiten representar valores identificables de forma descriptiva, esto incluye nombres de personas, direcciones, etc. Es posible representar números como alfanuméricos, pero estos pierden su propiedad matemática, es decir no es posible hacer operaciones con ellos. Este tipo de datos se representan encerrados entre comillas.

Ejemplo: "Politécnico Colombiano Jaime Isaza Cadavid"
"2007"

2. EXPRESIONES

Las expresiones son combinaciones de constantes, variables, símbolos de operación, paréntesis y nombres de funciones especiales. Por ejemplo:

a+(b + 3)/c

Cada expresión toma un valor que se determina tomando los valores de las variables y constantes implicadas y la ejecución de las operaciones indicadas.

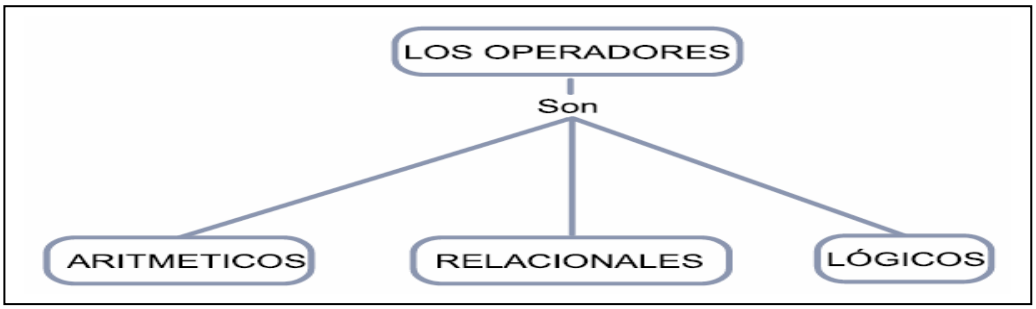
Una expresión consta de operadores y operandos. Según sea el tipo de datos que manipulan, se clasifican las expresiones en:

- Aritméticas
- Relaciónales
- Lógicas

3. OPERADORES Y OPERANDOS

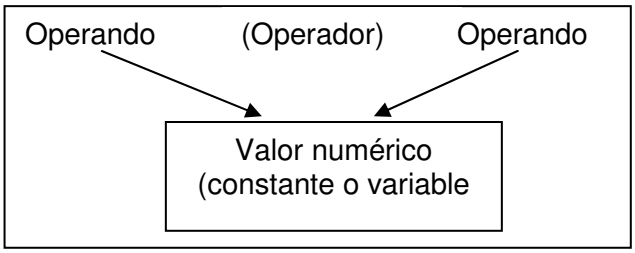
➢ **Operadores**: Son elementos que relacionan de forma diferente, los valores de una o mas variables y/o constantes (operandos). Es decir, los operadores nos permiten manipular valores.

Tipos de operadores	Aritméticos Relacionales Lógicos



- **Operadores Aritméticos**: Los operadores aritméticos permiten la realización de operaciones matemáticas con los valores (variables y constantes) para dar como resultado un número.

  Los operadores aritméticos pueden ser utilizados con tipos de datos enteros o reales. Si ambos son enteros, el resultado es entero; si alguno de ellos es real, el resultado es real.



Operadores Aritméticos		
Símbolo	Significado	Prioridad
^ o **	Potencia	↓
*, /, %, \	Multiplicación, División, Modulo (residuo de la división entera), Cociente	
+, -	Suma, Resta	

Ejemplos:

Expresión	Resultado
7 / 2	3.5
12 % 7	5
4 + 2 * 5	14

Prioridad de los Operadores Aritméticos

- Todas las expresiones entre paréntesis se evalúan primero. Las expresiones con paréntesis anidados se evalúan de dentro a fuera, el paréntesis más interno se evalúa primero.
- Dentro de una misma expresión los operadores se evalúan en el siguiente orden.
  1.- ^ Exponenciación
  2.- *, /, % Multiplicación, división, modulo.
  3.- +, - Suma y resta.
- Los operadores en una misma expresión con igual nivel de prioridad se evalúan de izquierda a derecha.

Ejemplos:

4 + 2 * 5 = 14	
23 * 2 / 5 = 9.2	46 / 5 = 9.2
3 + 5 * (10 - (2 + 4)) = 23	3 + 5 * (10 - 6) = 3 + 5 * 4 = 3 + 20 = 23
3.5 + 5.09 - 14.0 / 40 = 5.09	3.5 + 5.09 - 3.5 = 8.59 - 3.5 = 5.09
2.1 * (1.5 + 3.0 * 4.1) = 28.98	2.1 * (1.5 + 12.3) = 2.1 * 13.8 = 28.98

- **Operadores Relaciónales:**

- Se utilizan para establecer una relación entre dos valores.
- Compara estos valores entre sí y esta comparación produce un resultado de verdad o falsedad (verdadero o falso).
- Los operadores relaciónales comparan valores del mismo tipo (numéricos o cadenas)
- Tienen el mismo nivel de prioridad en su evaluación.
- Los operadores relaciónales tiene menor prioridad que los aritméticos.

Operadores relaciónales	
Operador	Significado
>	Mayor que
<	Menor que
> =	Mayor o igual que
< =	Menor o igual que
< >	Diferente
=	Igual

Ejemplos:

Si a = 10 b = 20 c = 30

a + b > c	Falso
a - b < c	Verdadero
a - b = c	Falso
a * b < > c	Verdadero

.V. < 30 (no es lógico porque tiene diferentes operandos).

- **Operadores Lógicos:**

- Estos operadores se utilizan para establecer relaciones entre valores lógicos.
- Estos valores pueden ser resultado de una expresión relacional.

Operadores Lógicos		
Operador	Significado	Prioridad
Not	Negación	↓
And	Y (lógica o conjunción)	
Or	O (lógica) o disyunción	

Operador And

Operando1	Operador	Operando2	Resultado
.V.	AND (^)	.V.	.V.
.V.	AND (^)	.F.	.F.
.F.	AND (^)	.V.	.F.
.F.	AND (^)	.F.	.F.

Operador Or

Operando1	Operador	Operando2	Resultado
.V.	OR (∨)	.V.	.V.
.V.	OR (∨)	.F.	.V.
.F.	OR (∨)	.V.	.V.
.F.	OR (∨)	.F.	.F.

Operador Not

Operando	Resultado
.V.	.F.
.F.	.V.

Ejemplos:

(a < b) and (b < c).

(10<20) and (20<30)
T and T → T

T equivalente a V

Prioridad de los Operadores en General

1.	( )
2.	^
3.	*, /, %, Not
4.	+,
5.	>, <, > =, < =, < >, =, Or

Ejemplos:

a = 10 b = 12 c = 13 d =10

1) ((a > b) or (a < c)) and ((a = c) or (a > = b))

(F or T) and (F or F)

T and F

F

2) ((a > = b) or (a < d)) and ((a > = d) and (c > d))

(F or F) and (T and T)

F and T

F

3) not (a = c) and (c > b)

not(F) and T

T and T

T

4. IDENTIFICADORES

Los identificadores representan los datos de un programa (constantes, variables, tipos de datos). Un identificador es una secuencia de caracteres que sirve para identificar una posición en la memoria de la computadora, que nos permite acceder a su contenido.

Ejemplo:

Nombre
Num_hrs
Calif2

Reglas para formar un Identificador

- Debe comenzar con una letra (A a Z, mayúsculas o minúsculas) y no deben contener espacios en blanco.
- Letras, dígitos y caracteres como la subraya (_) están permitidos después del primer carácter.
- La longitud de identificadores depende del lenguaje de programación.

5. TIPOS DE CAMPOS: CONSTANTES Y VARIABLES

- **Constante**: Una constante es un dato numérico o alfanumérico que no cambia durante la ejecución del programa.

  Ejemplo:

  pi = 3.1416

- **Variable:** Es un espacio en la memoria de la computadora que permite almacenar temporalmente un dato durante la ejecución de un proceso, su contenido puede cambia durante la ejecución del programa. Para poder reconocer una variable en la memoria de la computadora, es necesario darle un nombre con el cual podamos identificarla dentro de un algoritmo.

  Ejemplo:

  área = pi * radio ^ 2

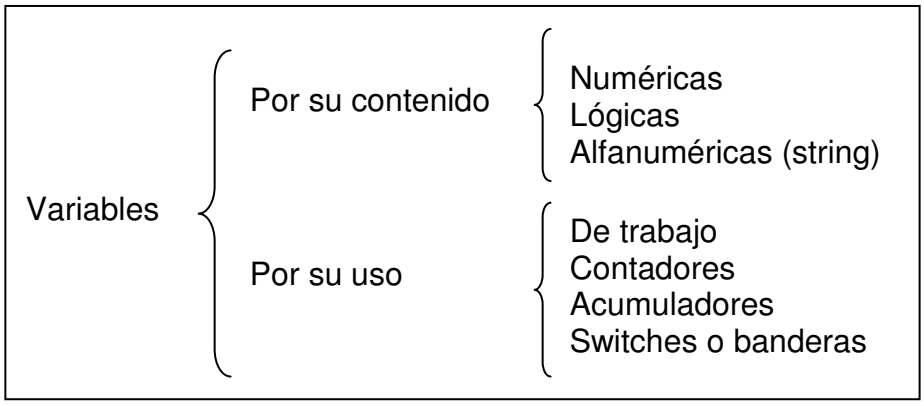
  Las variables son:

  radio y área

  La constate es:

  pi

Clasificación de las Variables



Por su Contenido.

- **Variable Numéricas:** Son aquellas en las cuales se almacenan valores numéricos, positivos o negativos, es decir almacenan números del 0 al 9, signos (+ y -) y el punto decimal. Ejemplo:

iva=0.16 pi=3.1416 costo=2500

- **Variables Lógicas:** Son aquellas que solo pueden tener dos valores (verdadero o falso) estos representan el resultado de una comparación entre otros datos.

- **Variables Alfanuméricas:** Esta formada por caracteres alfanuméricos (letras, números y caracteres especiales).

Ejemplo:

letra='a' apellido="lopez" direccion="Av. Oriental #45-54"

Por su uso.

- **Variables de Trabajo:** Variables que reciben el resultado de una operación matemática completa y que se usan normalmente dentro de un programa. Ejemplo:

suma=a+b/c

- **Contadores:** Se utilizan para llevar el control del número de ocasiones en que se realiza una operación o se cumple una condición. Se incrementa o disminuye en una cantidad constante, así:

Contador = Contador + <constante>

Los contadores se deben inicializar (darle un valor inicial) antes de utilizarlos debido a que algunos compiladores al asignarles espacio de memoria los llena con unos números indefinidos llamados "basura".

➢ **Acumuladores:** Forma que toma una variable y que sirve para llevar la suma acumulativa de una serie de valores que se van leyendo o calculando progresivamente. Se incrementan en una cantidad variable resultante de algún proceso, así:

Acumulador = Acumulador + <expresión>

En donde "expresión" puede ser una simple variable o varias variables operadas en alguna forma.

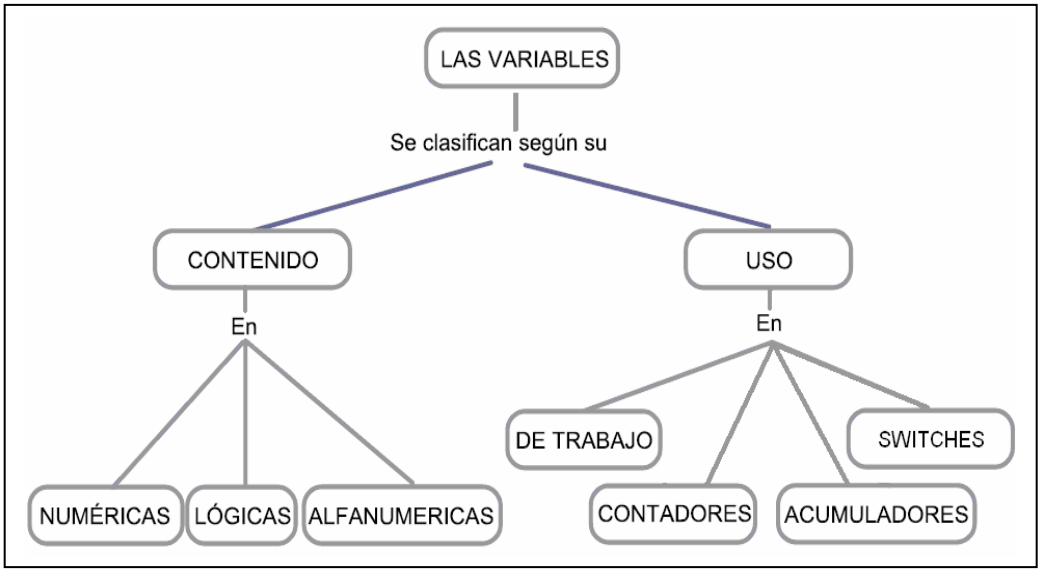
Por la misma razón de los contadores, los acumuladores también deben ser inicializados.

➢ **Switches o banderas:**

Una variable tipo switche o bandera es aquella que almacena generalmente uno de dos valores excluyentes entre sí. Los valores los define el programador y son normalmente 0 y 1 'S' y 'N' 'F' y 'V' "SI" y "NO" "VERDADERO" y "FALSO"

Los switches o banderas se utilizan para:

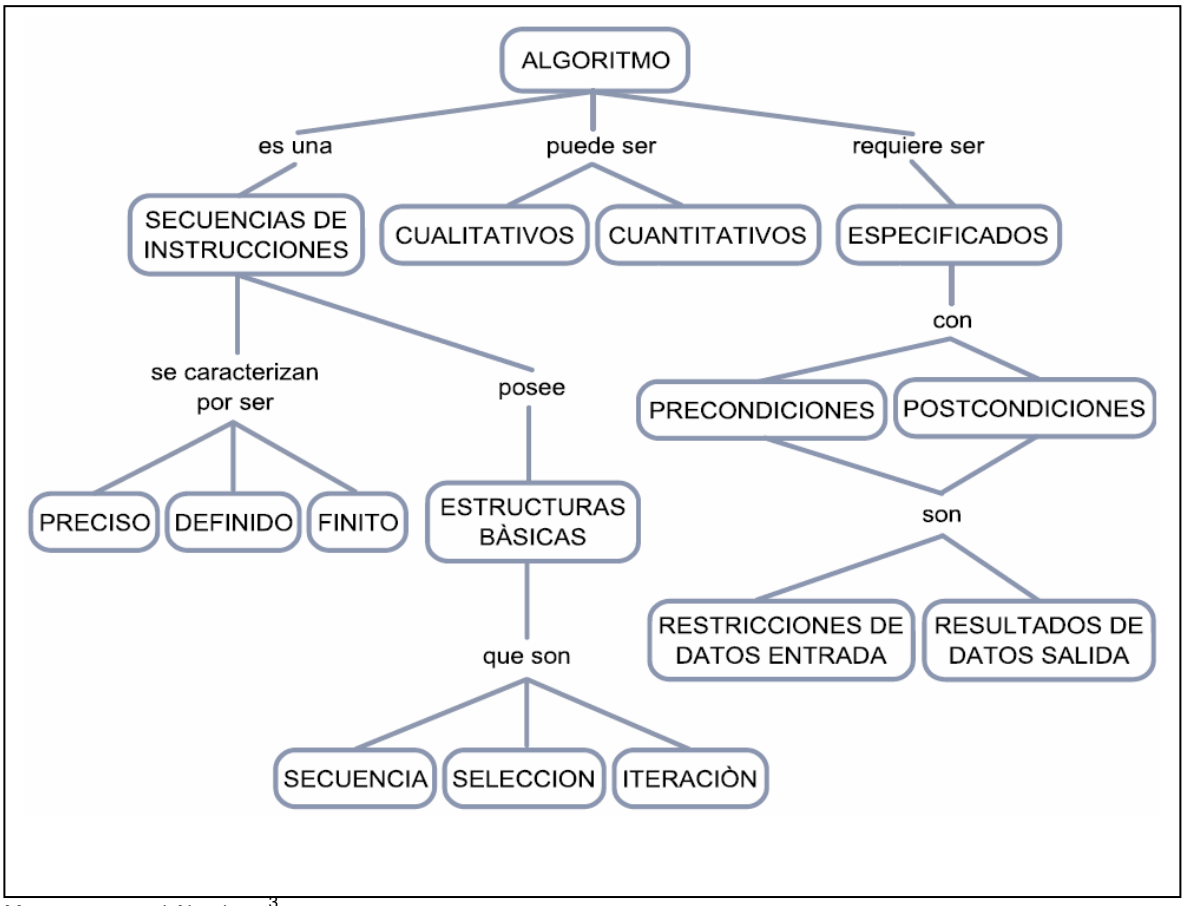
a. controlar un ciclo cuando no se puede controlar con otro tipo de variable.
b. para indicar que ya se consiguió u ocurrió algo que se esperaba en el algoritmo.
c. para alternar los caminos dentro de un ciclo.



Mapa conceptual clasificación de las variables[2]

[2] Tomado del libro "Fundamentos de programación con énfasis en análisis y metodología para trabajo en equipos efectivos"

6. DEFINICIÓN DE ALGORITMO



Mapa conceptual Algoritmo [3]

Algoritmo se deriva de la traducción al latín de la palabra árabe alkhowarizmi, nombre de un matemático y astrónomo árabe que escribió un tratado sobre manipulación de números y ecuaciones en el siglo IX.

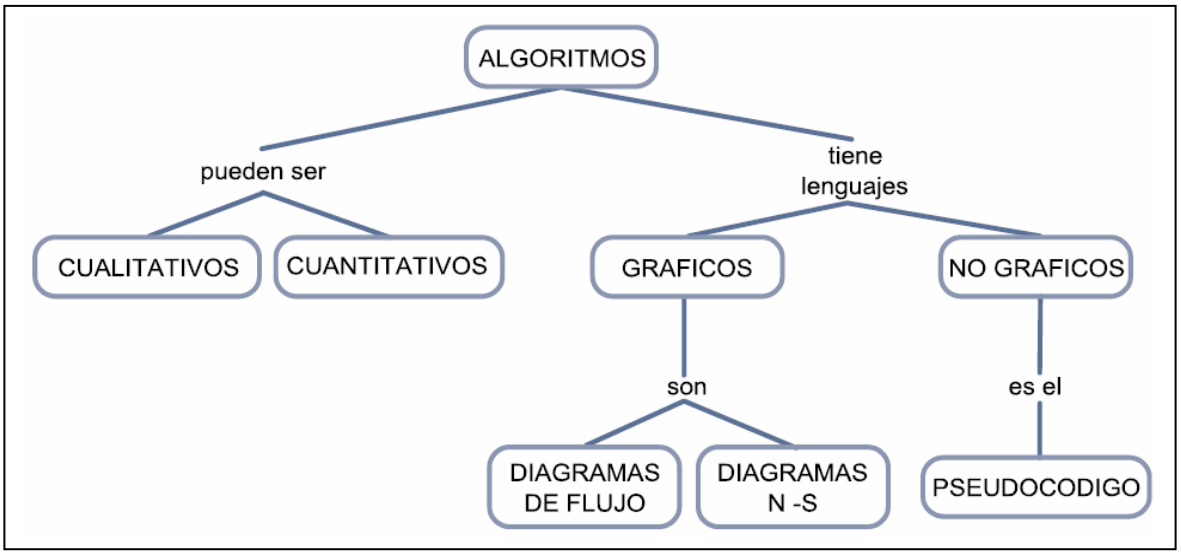
Un **algoritmo** es una serie de pasos organizados que describe el proceso que se debe seguir, para dar solución a un problema específico.

Tipos de Algoritmos.

- Cualitativos o procedimientos: En ellos se describen los pasos utilizando palabras.
- Cuantitativos: En ellos se utilizan cálculos numéricos para definir los pasos del proceso.

[3] Tomado del libro "Fundamentos de programación con énfasis en análisis y metodología para trabajo en equipos efectivos"

7. FORMAS DE REPRESENTACIÓN DE UN ALGORITMO



Mapa conceptual representacion de un Algoritmo [4]

La representación de un algoritmo consta de una serie de símbolos y reglas que se utilizan para describir de manera explícita un proceso. Hay dos formas fundamentales de representar un algoritmo:

- **Gráficos**: Es la representación gráfica de las operaciones que realiza un algoritmo (diagrama de flujo y diagramas rectangulares).

- **No Gráficos**: Representa en forma descriptiva las operaciones que debe realizar un algoritmo (pseudocódigo).

Las dos herramientas utilizadas comúnmente para representar algoritmos son:

Diagramas (de flujo y rectangular)
Pseuducódigo

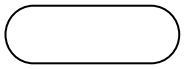
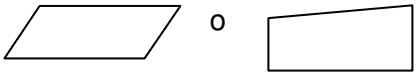
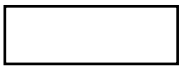
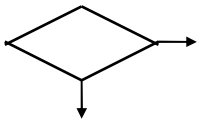
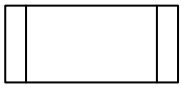
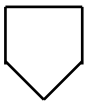
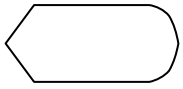
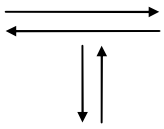
Diagrama de Flujo.

Es la representación gráfica de un algoritmo. También es la representación detallada en forma gráfica de como se realizan los pasos en la computadora para producir resultados.

Esta representación gráfica se da cuando varios símbolos (que indican diferentes instrucciones en la computadora), se relacionan entre si mediante líneas que indican el orden en que se deben ejecutar los procesos.

[4] Tomado del libro "Fundamentos de programación con énfasis en análisis y metodología para trabajo en equipos efectivos"

Los símbolos utilizados han sido normalizados por el instituto norteamericano de normalización (ANSI).

SÍMBOLO	DESCRIPCIÓN
	Indica el inicio y el final de nuestro diagrama de flujo.
	Indica la entrada de datos
	Símbolo de proceso y nos indica la asignación de un valor en la memoria y/o la ejecución de una operación aritmética.
	Símbolo de decisión indica la realización de una comparación de valores.
	Se utiliza para representar los subprogramas.
	Conector dentro de página. Representa la continuidad del diagrama dentro de la misma página.
	Conector fuera de página. Representa la continuidad del diagrama en otra página.
	Indica la salida de información por impresora.
	Indica la salida de información en la pantalla o monitor.
	Líneas de flujo o dirección. Indican la secuencia en que se realizan las operaciones.

Recomendaciones para el diseño de Diagramas de Flujo

- ✓ Se deben se usar solamente líneas de flujo horizontales y/o verticales.
- ✓ Se debe evitar el cruce de líneas utilizando los conectores.
- ✓ Se deben usar conectores solo cuando sea necesario.
- ✓ No deben quedar líneas de flujo sin conectar.
- ✓ Se deben trazar los símbolos de manera que se puedan leer de arriba hacia abajo y de izquierda a derecha.
- ✓ Todo texto escrito dentro de un símbolo deberá ser escrito claramente, evitando el uso de muchas palabras.

Pseudocódigo.

Mezcla de lenguaje de programación y español (o inglés o cualquier otro idioma) que se emplee dentro de la programación estructurada, para realizar el diseño de un programa. El pseudocódigo se puede definir como un lenguaje de especificaciones de algoritmos.

Es la representación narrativa de los pasos que debe seguir un algoritmo para dar solución a un problema determinado. El pseudocódigo utiliza palabras que indican el proceso a realizar.

Ventajas de utilizar un Pseudocódigo a un Diagrama de Flujo

- Ocupa menos espacio en una hoja de papel
- Permite representar en forma fácil operaciones repetitivas complejas
- Es muy fácil pasar de pseudocódigo a un programa en algún lenguaje de programación.
- Si se siguen las reglas se puede observar claramente los niveles que tiene cada operación.

Diagramas estructurados o rectangulares (Nassi-Schneiderman).

El diagrama estructurado N-S también conocido como diagrama de chapin es como un diagrama de flujo en el que se omiten las flechas de unión y las cajas son contiguas. Las acciones se pueden escribir en cajas sucesivas y como en los diagramas de flujo, se pueden escribir diferentes acciones en una caja. Un algoritmo se represente en la siguiente forma:

Inicio
Accion1
Accion2
. . .
Fin

Ejemplo:

```
Inicio

   Leer
    Nombre,Hrs,Precio
   Calcular
     Salario = Hrs * Precio
   Calcular
     Imp = Salario*  0.15
   Calcular
      Neto = Salario + Imp
   Escribir
     Nombre, Imp, SNeto

Fin
```

ACTIVIDADES

1) Responder al siguiente cuestionario.

1 La representación de un algoritmo se define como
- **A.** *U*na serie de códigos y reglas que se utilizan para describir de manera explícita un proceso..
- **B.** *U*na serie de símbolos y reglas que se utilizan para describir de manera explícita un proceso..
- **C.** *U*na serie de símbolos y reglas que se utilizan para describir de manera implícita un proceso..

2 Una variable se refiere a:
- **A.** Un espacio en la memoria de la computadora que permite almacenar temporalmente un dato durante la ejecución de un proceso, su contenido puede cambiar durante la ejecución del programa. .
- **B.** Un espacio en la memoria de la computadora que permite almacenar permanentemente un dato durante la ejecución de un proceso,
- **C.** Un espacio en la memoria de la computadora que permite almacenar un dato durante la ejecución de un proceso, su contenido no puede cambia.

3 La definición siguiente : Es un diagrama de flujo en el que se omiten las flechas de unión y las cajas son contiguas se le denomina a:
- **A.** El diagrama estructurado.
- **B.** El diagrama natural.
- **C.** El diagrama jerárquico.

4 El único lenguaje que la computadora entiende se denomina.
- **A.** Lenguaje de programación.
- **B.** Lenguaje objeto.
- **C.** Lenguaje fuente.

5 Cual es el concepto de compilación.
- **A.** Es la persona encargada del sistema de base de datos
- **B.** Es el análisis del compilador a unas instrucciones para detectar si están bien escritas, si no existen errores se transcribe a lenguaje de maquina
- **C.** Es la orden para que el compilador analice las instrucciones del programa fuente para detectar si están bien escritas o no.

2) Evalúe las siguientes expresiones según la prioridad de los operadores

- **A.** *15 * 14 – 3 * 7*
- **B.** *–4 * 5 * 2*
- **C.** *(24 + 2 * 6) \ 4*
- **D.** *3 + 4 * (8 * (4 - (9 + 3) DIV 6))*
- **E.** *7 * 10 – 5 % 3 * 4* + 9

Dado que: A = 20 B = -10 C = 15 D = 17

A. A >= D **or** NOT (A < 10) **and** A = b

B. **not**(VERDADERO **or** A < C **and** D <> B)

C. ((D + A / B) > -7) **or** A **=** D **and** 3 <= C

ESTRATEGIAS METODOLÓGICAS

- ✓ Presentación de la unidad por parte del profesor.
- ✓ Explicación de los tipos de campos que se puede tener en la memoria del computador
- ✓ Explicación de las clases de variables.
- ✓ Explicación de los tipos de expresiones que puede haber en un algoritmo.
- ✓ Explicación de los operadores que se puede presentar en una expresión.
- ✓ Realización de ejercicios con expresiones.
- ✓ Realización de ejercicios de conversión de expresiones algebraicas en expresiones algorítmicas.

RECURSOS

- ✓ Humanos: profesor y alumnos.
- ✓ Institucionales: Salón de clases.
- ✓ Materiales: Módulo guía y tablero

INDICADORES DE EVALUACIÓN

- ✓ Evaluación escrita sobre la unidad

CRITERIOS DE EVALUACIÓN

- ✓ Enumerar los tipos de campos que se puede tener en la memoria del computador
- ✓ Enumerar las clases de variables.
- ✓ Enumerar los tipos de expresiones que puede haber en un algoritmo.
- ✓ Identificar los operadores que se puede presentar en una expresión.
- ✓ Realizar ejercicios con expresiones.
- ✓ Realizar ejercicios de conversión de expresiones algebraicas a expresiones algorítmicas.

UNIDAD 3

METODOLOGÍA PARA LA SOLUCIÓN DE PROBLEMAS POR MEDIO DEL COMPUTADOR

INTRODUCCIÓN

Existen una serie de principios que gobiernan el comportamiento de los computadores y programas; los cuales establecen la forma de abordar un problema cuando se pretende resolverlo a partir de un computador. Este conjunto de principios recibe el nombre de *Resolución de problemas algorítmicos*.

Un problema algorítmico es cualquier problema conceptual o práctico cuya solución puede expresarse mediante un algoritmo.

La resolución de problemas a partir del computador conduce a la escritura de un programa y a la ejecución de éste; pero para poder abordar el proceso detalladamente se hace necesario cumplir con una fase preliminar que no hace parte del proceso de resolución como tal, aunque muchos autores la incluyen en éste.[3] Esta fase se refiere al entendimiento del problema. No se puede pretender platear una solución a un problema dado, si no se ha entendido éste en toda su dimensión.

JUSTIFICACIÓN

El hacer uso de una metodología para la resolución de problemas, se convierte en el factor diferenciador entre el programador empírico y la persona que realiza un estudio tendiente a la solución de problemas.

Se ha demostrado que se obtienen mejores resultados en la implantación de una solución a un problema dado, cuando esta, es el resultado de la secuenciación adecuada de todos los pasos propuestas como metodología para la solución de problemas. Por lo anterior se hace necesario realizar un acercamiento a cada uno de los pasos estipulados para tal proceso.

OBJETIVO GENERAL

Comprender como se realiza la resolución de problemas identificando etapas, estrategias y herramientas lógico matemáticas para aplicarlas a la programación de computadores.

[5] Tomado del libro "Fundamentos de programación con énfasis en análisis y metodología para trabajo en equipos efectivos"

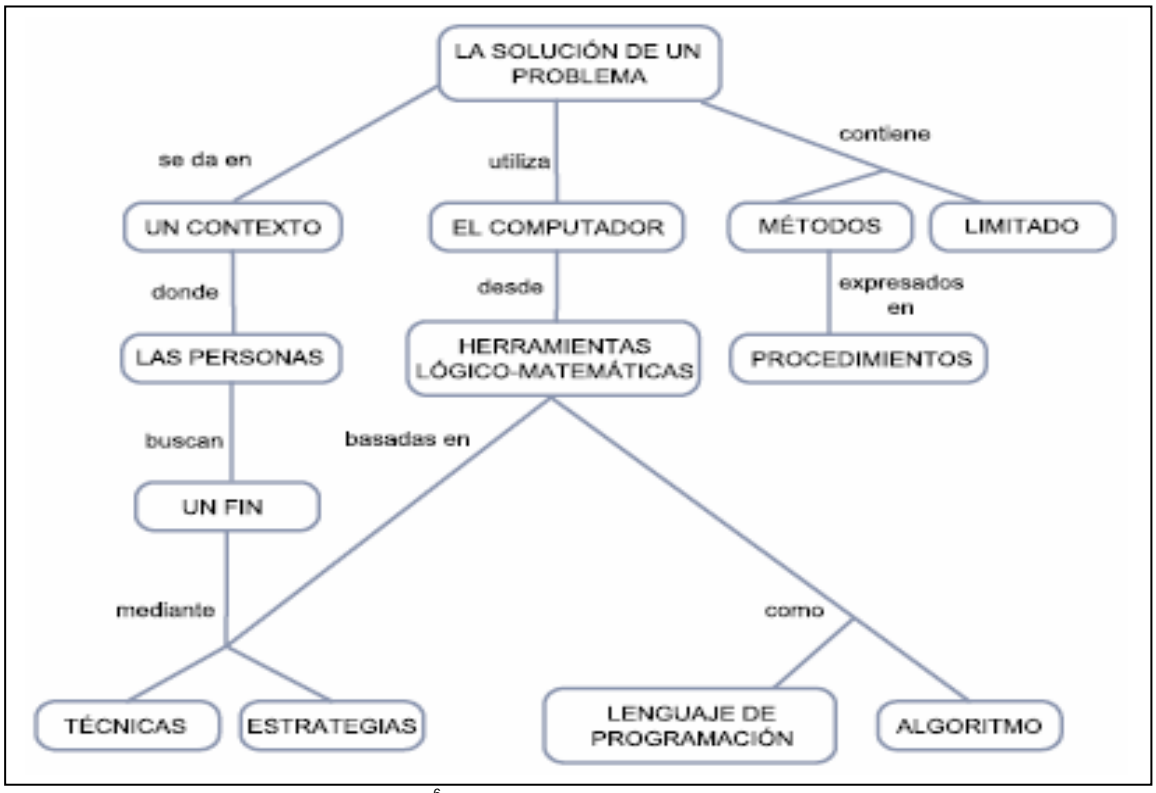
OBJETIVOS ESPECÍFICOS

1. Entender como sirven de apoyo las herramientas lógico matemáticas para resolver problemas.
2. Fomentar la aplicación de diversas estrategias para la resolución de problemas.
3. Identificar las fases requeridas para resolver un problema y como se desarrollan y aplican.

CONTENIDO

1. Introducción a la solución de problemas.
2. Pasos para la solución de problemas por medio del Computador.

1. INTRODUCCION A LA SOLUCIÓN DE PROBLEMAS.



Mapa conceptual Solución de un problema[6]

[6] Tomado del libro "Fundamentos de programación con énfasis en análisis y metodología para trabajo en equipos efectivos"

2. Pasos Para La Solución De Problemas Por Medio De Un Computador

Para hacer un programa de computadora que solucione un problema del usuario, es necesario realizar una serie de pasos secuenciales, ellos son:

- *Definición del Problema*

Esta fase está dada por el enunciado del problema, el cual requiere una definición clara y precisa. Es importante que se conozca lo que se desea que realice la computadora; mientras esto no se conozca del todo no tiene mucho caso continuar con la siguiente etapa.

- *Análisis del Problema*

Una vez que se ha comprendido lo que se desea de la computadora, es necesario definir:

- Los datos de entrada.
- Cual es la información que se desea producir (salida)
- Los métodos y fórmulas que se necesitan para procesar los datos.

Una recomendación muy práctica es el que nos pongamos en el lugar de la computadora y analicemos qué es lo que necesitamos que nos ordenen y en qué secuencia para producir los resultados esperados.

- *Diseño del Algoritmo*

Las características de un buen algoritmo son:

- Debe tener un punto particular de inicio.
- Debe ser definido, no debe permitir dobles interpretaciones.
- Debe ser general, es decir, soportar la mayoría de las variantes que se puedan presentar en la definición del problema.
- Debe ser finito en tamaño y tiempo de ejecución.

- *Prueba de escritorio*

Permite detectar los posibles errores que pueda haber cometido el programador en la elaboración del algoritmo y consiste en darle valores a los datos de entrada, hacer un seguimiento del algoritmo y comparar los resultados arrojados con los esperados realmente. En la prueba de escritorio se detectan errores de lógica.

- *Codificación*

La codificación es la operación de escribir la solución del problema (de acuerdo a la lógica del diagrama de flujo o pseudocódigo), en una serie de instrucciones detalladas, en un código reconocible por la computadora, la serie de instrucciones detalladas se le conoce como **código fuente**, el cual se escribe en un lenguaje de programación o lenguaje de alto nivel.

- *Trascripción*

Consiste en digitar el programa y guardarlo en un medio de almacenamiento. El programa trascrito se llama **programa fuente**. Este programa posteriormente se podrá modificar o depurar de cualquier forma.

- *Compilación*

Consiste en hacer (dar la orden para) que el compilador analice una a una las instrucciones del programa fuente para detectar si están bien escritas o no, de acuerdo a las reglas (sintaxis) del lenguaje. Si existen errores se producirá un listado de éstos indicando su localización y causa. Si no existen errores, el programa se traduce al **lenguaje de máquina**, único lenguaje que la computadora entiende, lo cual da origen a lo que se denomina **programa objeto**. En esta fase se le asigna dirección de memoria a todos los campos y además se le da valor a los campos constantes.

- *Ejecución*

Consiste en dar la orden para que sean ejecutadas, una a una, las instrucciones del programa objeto y para ello es necesario darle valor a las variables de entrada a medida que el programa lo vaya pidiendo, de igual forma que se hizo en la prueba de escritorio. En esta fase se le da valores a los campos variables y se detectan los errores de ejecución.

- *Prueba y Depuración*

Los errores humanos dentro de la programación de computadoras son muchos y aumentan considerablemente con la complejidad del problema. El proceso de identificar y eliminar errores, para dar paso a una solución sin errores se le llama depuración.

La depuración o prueba resulta una tarea tan creativa como el mismo desarrollo de la solución, por ello se debe considerar con el mismo interés y entusiasmo. Resulta conveniente realizar una buena depuración, ya que de este trabajo depende el éxito de nuestra solución.

- *Documentación*

Es la guía o comunicación escrita en sus variadas formas, ya sea en enunciados, procedimientos, dibujos o diagramas.

A menudo un programa escrito por una persona, es usado por otra. Por ello la documentación sirve para ayudar a comprender o usar un programa o para facilitar futuras modificaciones (mantenimiento).

La documentación se divide en tres partes:

- Documentación Interna
- Documentación Externa
- Manual del Usuario

➢ Documentación Interna: Son los comentarios o mensaje que se añaden al código fuente para hacer mas claro el entendimiento de un proceso.

➢ Documentación Externa: Se define en un documento escrito los siguientes puntos:
  - Descripción del Problema
  - Nombre del Autor
  - Algoritmo (diagrama de flujo o pseudocódigo)
  - Diccionario de Datos
  - Código Fuente (programa)

➢ Manual del Usuario: Describe paso a paso la manera como funciona el programa, con el fin de que el usuario obtenga el resultado deseado.

- *Mantenimiento*

Se lleva a cabo después de terminado el programa, cuando se detecta que es necesario hacer algún cambio, ajuste o complementación al programa para que siga trabajando de manera correcta. Para poder realizar este trabajo se requiere que el programa esté correctamente documentado.

ACTIVIDADES

1) Responder al siguiente cuestionario.

1 Cuál de las fases en la solución de un problema, permite detectar los posibles errores que pueda haber cometido el programador en la elaboración del algoritmo:

A. Análisis.
B. Definición del problema.
C. Prueba de escritorio.
D. Diseño del algoritmo.

2 Se conoce como compilación el proceso en que:

A. Se convierte un programa de código objeto a código fuente
B. Proceso que analiza una a una las instrucciones del programa fuente para detectar si están bien escritas o no.
C. Proceso de documentar un programa
D. Proceso de depuración de errores

3 La fase que se lleva a cabo después de terminado el programa, cuando se detecta que es necesario hacer algún cambio, ajuste o complementación al programa para que siga trabajando de manera correcta, se conoce como:

A. Prueba de escritorio
B. Prueba y depuración
C. Codificación
D. Mantenimiento

4 En la fase de análisis se definen:

A. Diagramas y seudocódigos
B. Datos de entrada, salida y proceso
C. Operandos y operadores
D. Ninguno de los anteriores

5 El código fuente se escribe en:

E. Diagramas y seudocódigos
F. Un lenguaje de programación
G. Diagramas N-s
H. ninguno de los anteriores.

ESTRATEGIAS METODOLÓGICAS

- ✓ Presentación de la unidad por parte del profesor.
- ✓ Explicación de las diferentes etapas en la solución de de problemas
- ✓ Planteamiento de ejemplos modelos
- ✓ Plenarias de discusión

RECURSOS

- ✓ Humanos: profesor y alumnos.
- ✓ Institucionales: Salón de clases.
- ✓ Materiales: Módulo guía y tablero

INDICADORES DE EVALUACIÓN

- ✓ Conversatorio de discusión alrededor de las diferentes fases
- ✓ Participación individual.

CRITERIOS DE EVALUACIÓN

- ✓ Conocimientos individuales.
  El estudiante debe de estar en capacidad de identificar plenamente las diferentes fases que plantea la metodología de solución de problemas

UNIDAD 4

ESTRUCTURA SECUENCIAL

INTRODUCCIÓN

La estructura secuencial se caracteriza porque una instrucción sigue a otra en secuencia. Las tareas se suceden de tal modo que la salida de una es la entrada de la siguiente y así sucesivamente hasta el fin del proceso.

JUSTIFICACIÓN

El diseñar un algoritmo implica que éste debe ser lo más claro posible; estructurado de tal forma que su lectura facilite el entendimiento de éste.

Partiendo de lo anterior surgió la estandarización de la estructura de los lenguajes de programación sin importar su complejidad, lo que permitió definir que estos pueden ser construidos a partir de la combinación de tres estructuras básicas de control de flujo: Secuencial, Condicional y Repetitiva.

La estructura secuencial se convierte en el pilar de cualquier algoritmo, pues se garantiza que un programa basado en ejecución secuencial, siempre ejecutará las mismas acciones, a menos que se implemente una de las otras estructuras.

OBJETIVO GENERAL

Aprender a emplear la estructura secuencial, haciendo uso correcto de las instrucciones de lectura, asignación y escritura.

OBJETIVOS ESPECÍFICOS

1. Hacer uso correcto de las diferentes formas de asignación.
2. Aprender como se realiza la lectura de datos.
3. Identificar como se realiza la escritura de datos y su correcta escritura.

CONTENIDO

1. CONCEPTO DE ENTRADA ASIGNACIÓN Y SALIDA

Entrada: La entrada o lectura, consiste en recibir desde un dispositivo de entrada (p.ej. el teclado) un valor. Esta operación se representa en un seudocódigo como sigue:

Leer (a)

Leer (a, b) *(para lectura de varias variables en una sola instrucción)*

Donde “a” y “b” son las variables que recibirán los valores

Asignación: La asignación consiste, en el paso de valores o resultados a una zona de la memoria. Dicha zona será reconocida con el nombre de la variable que recibe el valor. La asignación se puede clasificar de la siguiente forma:

- Asignación Simples: Consiste en pasar un valor constate a una variable

  Ejemplo a = 15

- Asignación por contador: Consiste en usarla como un verificador del número de veces que se realiza un proceso.

  Ejemplo a = (a + 1)

- Asignación por Acumulación: Consiste en usarla como un sumador en un proceso

  Ejemplo a = (a + b)

- De trabajo: Donde se puede recibir el resultado de una operación matemática que involucre muchas variables

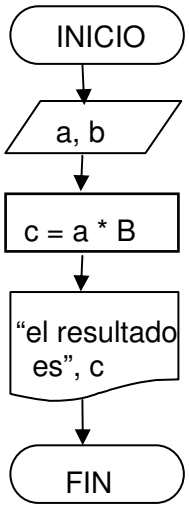
  Ejemplo a = c + b * 2 / 4

Salida. La salida o escritura, consiste en mandar por un dispositivo de salida (por ejemplo monitor o impresora) un resultado o mensaje. Este proceso se representa en seudocódigo como sigue:

Imprima (“El resultado es:”)	*imprime un mensaje.*
Imprima (a)	*imprime el contenido de la variable a.*
Imprima (“El resultado es: ” , a)	*imprime un mensaje mas el contenido de la variable*

2. REPRESENTACIÓN DE LA ESTRCTURA SECUENCIAL

a. Representación por medio de un Diagrama de Flujo.



b. Representación por medio de un Diagrama Estructurado (Diagrama N-S)

Inicio
Leer a, b,
Calcular c = a * b
Escribir “el resultado es”, c
Fin

c. Representación por Medio de Seudocódigo.

```
Inicio
  Leer a, b
  C = (a*b)
  Imprima (“el resultado es”, c)
Fin
```

3. EJEMPLO 1

a. Definición

Hallar el área del círculo a partir de un radio determinado

b. Análisis

Datos de entrada

- *Radio R*

Dato de salida

- *Area A*

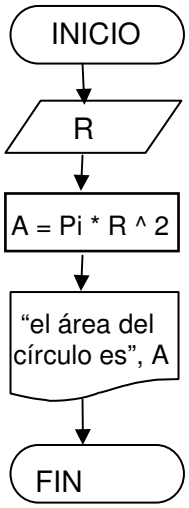
Datos de Proceso

- $A = Pi * R^2$
- *Pi es una constante = a 3.14159265*

c. Seudocódigo

```
Inicio
  Lea Radio
  Area = (Pi * Radio ^ 2)
  Imprima (“El área del círculo es”, Area)
Fin
```

d. Diagrama de flujo



e. Diagrama estructurado N-S

Inicio
Leer R
Calcular A = Pi * R ^ 2
Escribir "el área del circulo es", A
Fin

4. EJEMPLO 2

a. Definición

Un almacén que vende zapatos a un precio fijo desea conocer su utilidad en el total de ventas, partiendo del número de artículos vendidos día. Se conoce que la utilidad es del 35%

b. Análisis

Datos de entrada

- *Cantidad de Zapatos vendidos* *Cant*
- *Valor de los zapatos* *Vr*

Dato de salida

- *Utilidad* *U*

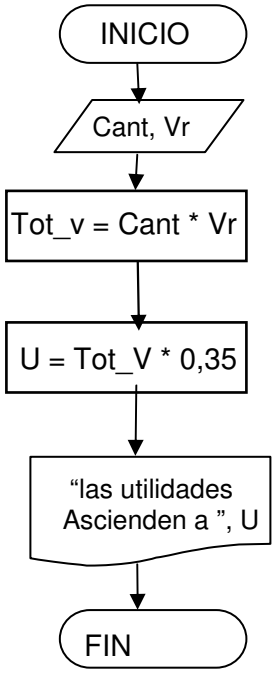
Datos de Proceso

- *Total ventas día* *Tot_v*
- *Tot_v = Cant * Vr*
- *U = Tot_v * 0,35*

c. Seudocódigo

```
Inicio
  Lea Cant, Vr
  Tot_v  = (Cant * Vr)
  U = Tot_V * 0,35
  Imprima ("Las utilidades ascienden a", U)
Fin
```

d. Diagrama de flujo



e. Diagrama estructurado N-S

Inicio
Leer Cant, vr
Calcular Tot_v = Cant * Vr
Calcular U = Tot_v * 0,35
Escribir "el área del circulo es", A
Fin

5. EJEMPLO 3

a. Definición

Un estudiante desea saber a partir de sus notas parciales, la definitiva en la asignatura de programación, teniendo en cuenta que ésta se califica de la siguiente forma: 1 seguimiento que equivale al 40%, un parcial que equivale al 20% un proyecto que equivale al 10% y un final que equivale al 30%.

b. Análisis

Datos de entrada

- Nota seguimiento N_s
- Nota Parcial N_p
- Nota proyecto N_py
- Nota examen final N_ef

Dato de salida

- Definitiva Def

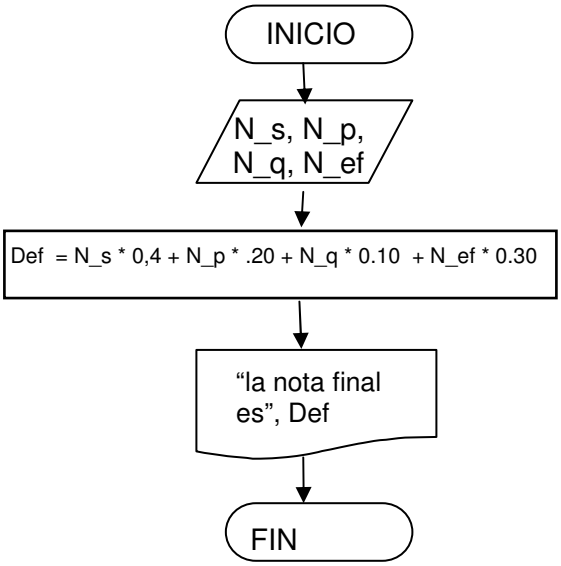
Datos de Proceso

- Def = N_s * 0,4 + N_p * .20 + N_q * 0.10 + N_ef * 0.30

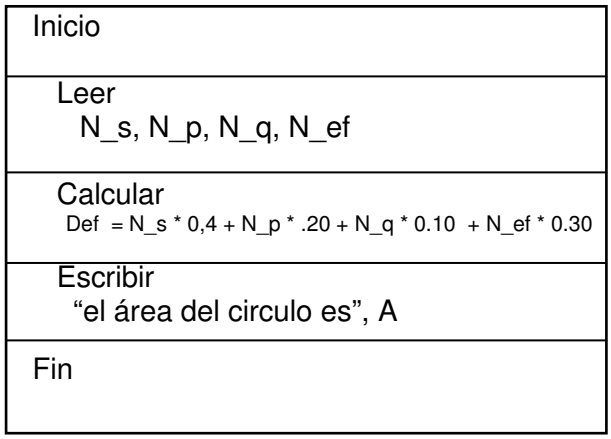
c. Seudocódigo

```
Inicio
  Lea  N_s,  N_p,  N_q,  N_ef

  Def  =( N_s * 0,4 + N_p * .20 + N_q * 0.10  + N_ef * 0.30)

  Imprima ("La nota definitiva es", Def)
Fin
```

e. Diagrama de flujo



e. Diagrama estructurado N-S



6. EJEMPLO 4

a. Definición

Un maestro desea saber que porcentaje de hombres y que porcentaje de mujeres hay en un grupo de estudiantes.

b. Análisis

Datos de entrada

- Total de hombres T_h
- Total de Mujeres T_m

Dato de salida

- Porcentaje de hombres P_h
- Porcentaje de mujeres P_m

Datos de Proceso

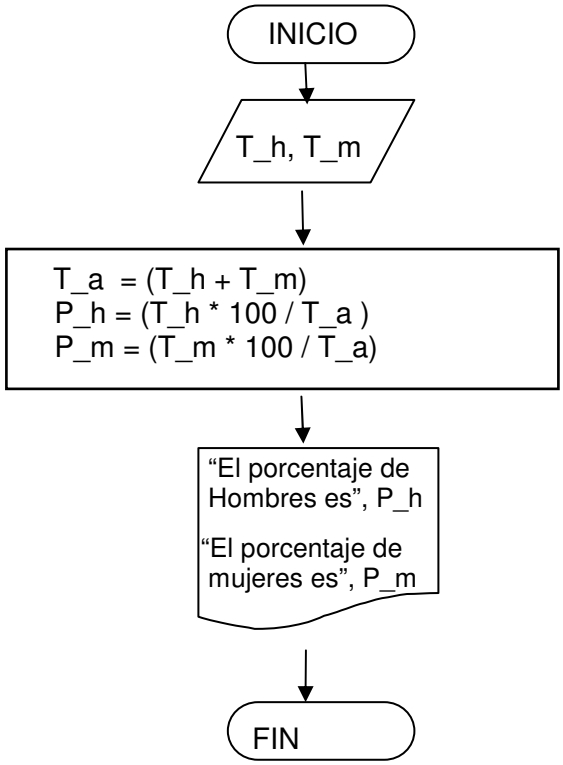
Para encontrar el porcentaje de hombres y de mujeres primero se debe obtener el total de alumnos

- Total de alumnos T_a
- T_a = T_h + T_m
- P_h = T_h / T_a * 100
- P_m = T_m / T_a * 100

c. Seudocódigo

```
Inicio
  Lea  T_h, T_m

  T_a  = (T_h + T_m)

  P_h = (T_h * 100 / T_a )

  P_m = (T_m * 100 / T_a)

  Imprima ("El porcentaje de hombres es: ", P_h)

  Imprima ("El porcentaje de mujeres es: ", P_m)

Fin
```

f. Diagrama de flujo



e. Diagrama estructurado N-S

Inicio
Leer T_h, T_m
Calcular T_a = (T_h + T_m) P_h = (T_h * 100 / T_a) P_m = (T_m * 100 / T_a)
Escribir "El porcentaje de hombres es", P_h "El porcentaje de mujeres es", P_m
Fin

7. EJEMPLO 5

a. Definición

Se desea saber cuantos pinos y cuantos cedros se pueden sembrar en un terreo que mide n cantidad de metros. El dueño ha establecido que sembrará el 35% del terreno en pinos y el 65% en cedros. Las normas de agricultura establecen que en 10 metros cuadrados se puede sembrar 4 pinos y en 15 metros cuadrados 5 cedros.

b. Análisis

Datos de entrada

- Número de metros a sembrar N_m

Dato de salida

- Cantidad de pinos a sembrar C_p
- Cantidad de cedros a sembrar C_c

Datos de Proceso

Para encontrar el número de pinos y cedros a sembrar, primero se debe obtener el porcentaje del terreno a sembrar por cada tipo de árbol.

- Porcentaje de pinos a sembrar P_p
- Porcentaje de cedros a sembrar P_c
- P_p = N_m * 0.35
- P_c = N_c * 0.65

Luego de conocer los porcentajes se aplica una regla de tres simple para conocer las cantidades, basándose en las normas establecidas por la agricultura

Si en 10 mtr se siembran 4 pinos
En P_p cuantos pinos se sembrarán

- C_p = P_p * 4 / 10

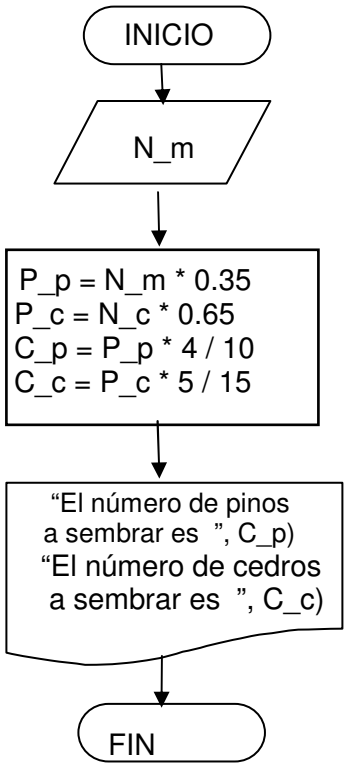
Si en 15 mtr se siembran 5 cedros
En P_c cuantos cedros se sembrarán

- C_c = P_c * 5 / 15

c. Seudocódigo

```
Inicio
  Lea  N_m

     P_p = N_m * 0.35

     P_c = N_c * 0.65

     C_p = P_p * 4 / 10

     C_c = P_c * 5 / 15

     Imprima ("El número de pinos a sembrar es  ", C_p)

     Imprima ("El número de cedros a sembrar es  ", C_c)
Fin
```

g. Diagrama de flujo



e. Diagrama estructurado N-S

Inicio
Leer N_m
Calcular P_p = N_m * 0.35 P_c = N_c * 0.65 C_p = P_p * 4 / 10 C_c = P_c * 5 / 15
Escribir “El número de pinos a sembrar es”, C_p “El número de cedros a sembrar es”, C_c
Fin

CRITERIOS DE EVALUACIÓN

Diseñe un algoritmo para cada uno de los problemas planteados.

1) **1.** Dada una cantidad en pesos, obtener la equivalencia en dólares, asumiendo que la unidad cambiaria es un dato desconocido.

2) La presión, el volumen y la temperatura de una masa de aire se relacionan por la formula:

masa ← (presión * volumen)/(0.37 * (temperatura + 460))

Encontrar la masa.

3) Calcular el número de pulsaciones que una persona debe tener por cada 10 segundos de ejercicio, si la formula es:

num. pulsaciones = (220 - edad)/10

4) Calcular el nuevo salario de un obrero si obtuvo un incremento del 25% sobre su salario anterior.

5) En un hospital existen tres áreas: Ginecología, Pediatría, Traumatología. El presupuesto anual del hospital se reparte conforme a la siguiente tabla:

Área	Porcentaje del presupuesto
Ginecología	40%
Traumatologia	30%
Pediatría	30%

Obtener la cantidad de dinero que recibirá cada área, para cualquier monto presupuestal.

ESTRATEGIAS METODOLÓGICAS

- ✓ Presentación de la unidad por parte del profesor.
- ✓ Elaboración y explicación magistral de los algoritmos por parte del docente.
- ✓ Prácticas en clase para el desarrollo de talleres

RECURSOS

- ✓ Humanos: profesor y alumnos y monitor.
- ✓ Institucionales: Salón de clases.
- ✓ Materiales: Módulo guía y tablero.

INDICADORES DE EVALUACIÓN

- ✓ Evaluación escrita sobre la unidad
- ✓ Talleres para desarrollar en grupos

CRITERIOS DE EVALUACIÓN

- ✓ Conocimientos individuales.
  El estudiante debe de estar en capacidad de solucionar problemas de complejidad básica, donde se demuestre el conocimiento de las instrucciones de lectura, asignación y escritura.

UNIDAD 5

ESTRUCTURAS CONDICIONALES O DE SELECCIÓN

INTRODUCCIÓN

Diariamente en nuestras vidas cotidianas tomamos decisiones sobre una serie de circunstancias que acontecen, por ejemplo, si queremos ir o no a un determinado lugar, si se dispone o no de dinero para ir de compras, si llueve o no, entre otras. Los condicionales permiten determinar si se realiza una determinada acción o no.

JUSTIFICACIÓN

No es posible imaginar un mundo donde no exista la posibilidad de decidir sobre algo, el solo hecho de imaginarlo nos trae a la mente un mundo estático e invariable, donde no hay la posibilidad de cambio y donde todo se puede preestablecer por anticipado, bueno, por fortuna las cosas no son así; vivimos en un mundo de cambio permanente y de situaciones que nos llevan a realizar acciones frente a las decisiones que tomemos. Los sistemas de información en las empresas no pueden ser estáticos, éstos deben permitir ejecutar una serie de acciones dependiendo los eventos que acontezcan, como por ejemplo, si un cliente esta en mora, si la existencia de un producto esta agotada, si el costo da perdida, entre otras.

OBJETIVO GENERAL

Aprender a emplear las diferentes estructuras condicionales permitiendo la representación de algoritmos que obedezcan a la toma de decisiones tendientes a solucionar un problema que plantea diferentes alternativas.

OBJETIVOS ESPECÍFICOS

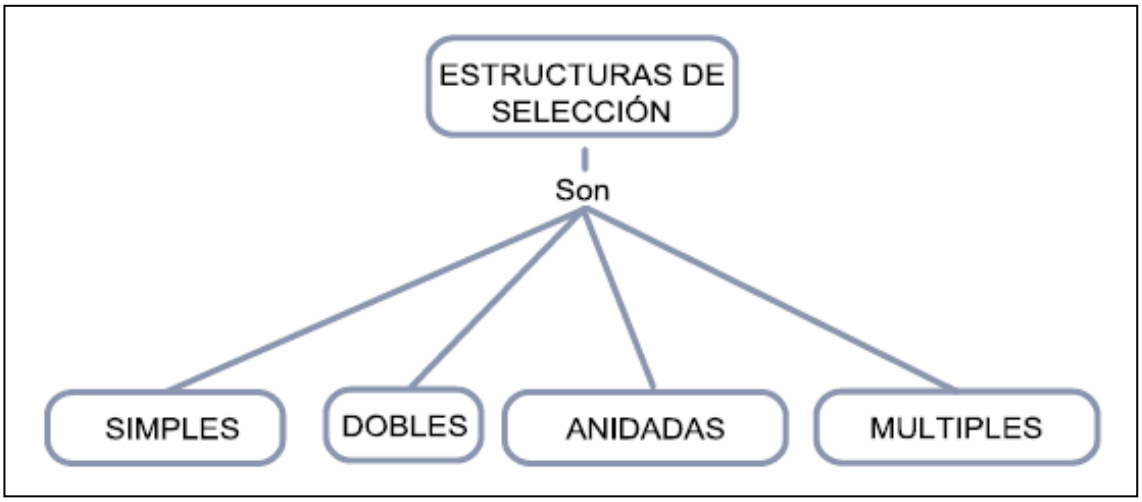
2. Aprender a emplear los condicionales simples.
3. Aprender a emplear los condicionales compuesto o anidados.
4. Aprender a elaborar condiciones compuestas empleando los operadores lógicos.
5. Aprender a realizar pruebas de escritorio a los ejercicios.

CONTENIDO

1. BASE TEÓRICA

Las estructuras de decisión o selección comparan una variable contra otro(s) valor(es), para que de acuerdo al resultado de esta comparación, se siga un curso de Acción dentro del programa. Cabe mencionar que la Comparación se puede hacer contra otra variable o contra una constante, según sea necesario. Existen cuatro tipos básicos:

simples, dobles, anidadas y múltiples; como se puede observar en siguiente mapa conceptual:

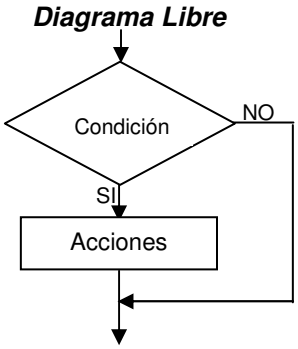


Mapa conceptual tipos de estructuras de decisión[7]

2. ESTRUCTURA CONDICIONAL SIMPLE (SI-ENTONCES)

Las estructuras condicionales simples permiten la toma de decisiones dentro de un programa. Dentro de una estructura selectiva, se incluye una expresión lógica (que devuelve un valor de verdad) que será la condición que se evalúa para definir la ruta que se seguirá dentro del programa. Si la expresión lógica (condición) es verdadera, entonces, se ejecutarán todas las sentencias dentro del bloque de la estructura condicional simple. Si la expresión lógica es falsa, no se ejecutará ninguna de estas sentencias.

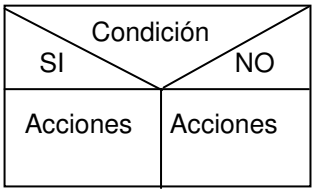
a. Representación por Medio de un Diagrama de Flujo Libre



[7] Tomado del libro "Fundamentos de programación con énfasis en análisis y metodología para trabajo en equipos efectivos"

b. Representación por Medio de un Diagrama de Flujo Estructurado

Diagrama Estructurado



c. Representación por Medio de Seudocódigo

Seudocódigo

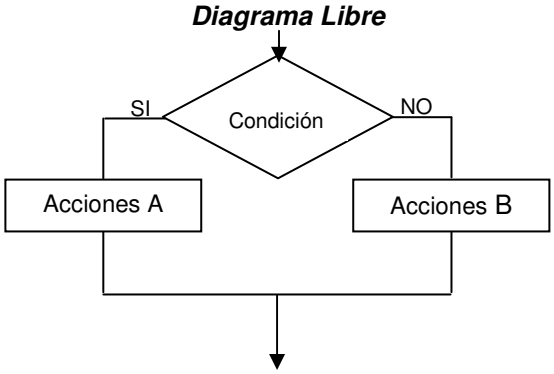
```
Si  (condición) entonces
     Acciones ……
FinSi
```

3. ESTRUCTURA CONDICIONAL DOBLE (SI-ENTONCES-SINO)

Otra forma de la estructura condicional, es la estructura selectiva doble, en la que se incluye una cláusula opcional si_no para establecer un grupo de acciones que se ejecutarán en caso de que la condición sea falsa.

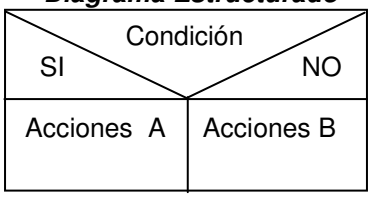
a. Representación por Medio de un Diagrama de Flujo Libe

Diagrama Libre



b. Representación por Medio de un Diagrama de Flujo Estructurado

Diagrama Estructurado



c. Representación por Medio de Seudocódigo

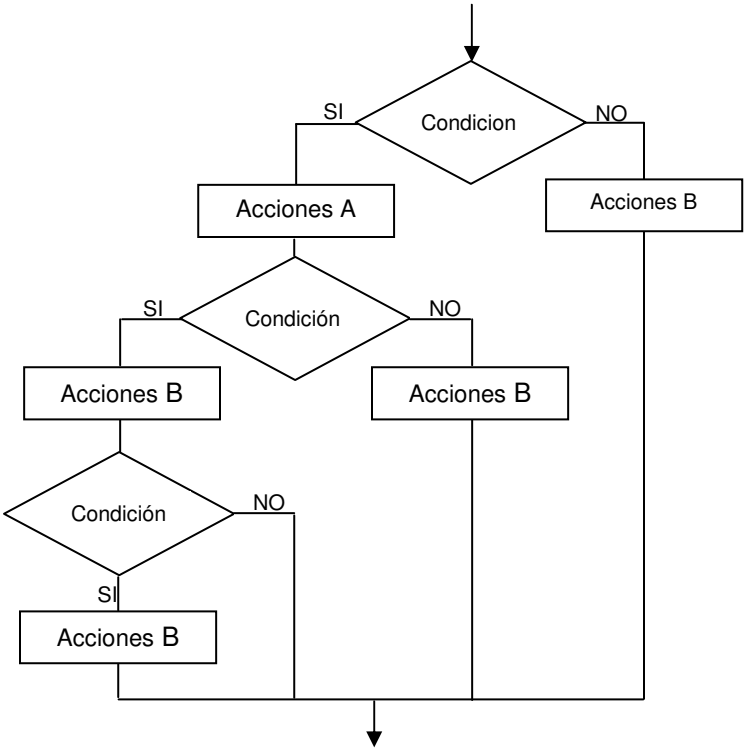
Seudocódigo

```
Si  (condición) entonces
     Acciones A
Sino
     Acciones B
FinSi
```

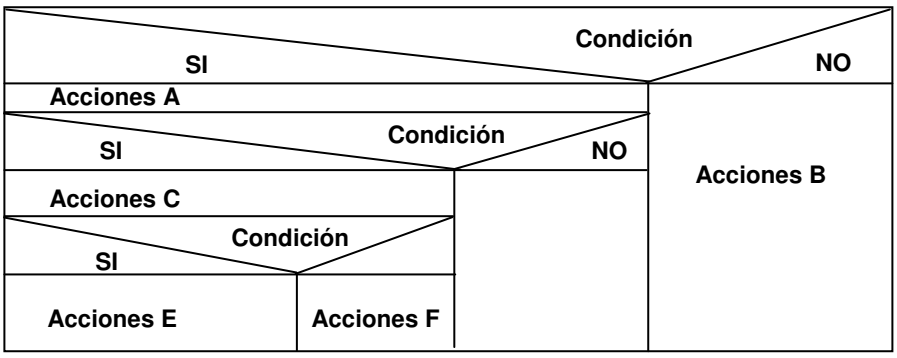
4. ESTRUCTURA CONDICIONAL ANIDADA

Las estructuras condicionales pueden anidarse, lo que significa que una estructura selectiva puede contener a su vez otra estructura selectiva, dentro de cualquiera de las secciones de una estructura condicional, sea simple o doble, puede incluirse otra estructura condicional que a su vez puede ser simple o doble y puede contener también dentro de ella, otras estructuras condicionales o selectivas.

a. Representación por Medio de un Diagrama de Flujo Libe



b. Representación por Medio de un Diagrama de Flujo Estructurado



c. Representación por Medio de Seudocódigo

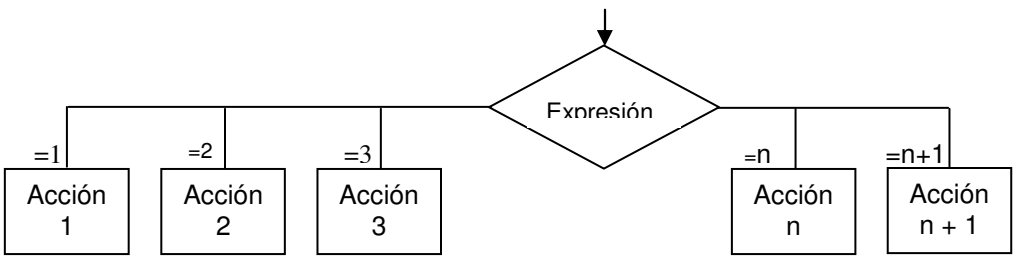
Seudocódigo

```
Si  (condición) entonces
    Acciones A
    Si  (condición) entonces
        Acciones C
        Si  (condición) entonces
            Acciones E
        FinSi
    Sino
        Acciones D
    FinSi
Sino
    Acciones B
FinSi
```

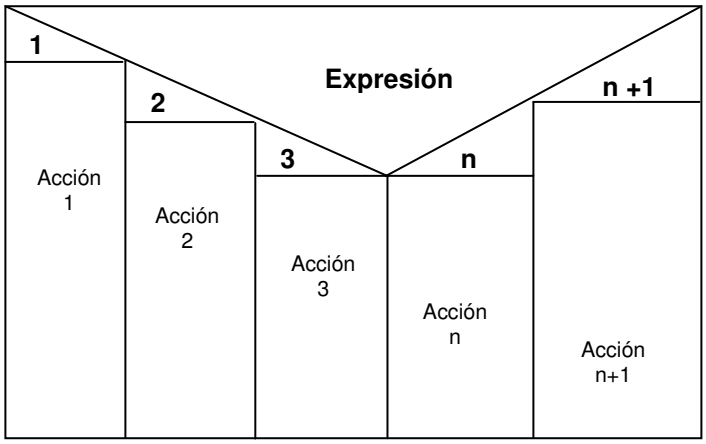
5. ESTRUCTURA MULTIPLE (SEGÚN CASO)

La estructura de selección múltiple segun_sea o casos, permite elegir una entre varias rutas posibles, evaluando para ello una expresión que puede tomar n valores distintos, 1, 2, 3, 4, ... , n.

a. Representación por Medio de un Diagrama de Flujo Libe



b. Representación por Medio de un Diagrama de Flujo Estructurado



c. Representación por Medio de Seudocódigo

Seudocódigo

```
Según (Expresión) Haga
        Caso Expr. = 1
                Acciones A
        Caso Expr.= 2
                Acciones B
        Caso Expr.= 3
                Acciones C
        Caso Expr.= n
                Acciones D
        Caso Expr.= n + 1
                Acciones E
        En_Otro_Caso
                Acciones F
FinSegún_Caso
```

6. EJEMPLO 1

a. Definición

Dados dos números A y B donde **A <> B**, determine cual es el mayor y menor e imprimirlos.

b. Análisis

Datos de entrada

- *A, B*

Dato de Proceso *(no hay formulas ni cálculos que hacer, sólo comparar)*

- *(A > B) mediante esta comparación podemos determinar cual de los dos es el mayor y el cual el menor.*

Datos de salida

- ***A*** *(mayor) y* ***B*** *(menor) ó* ***A*** *(menor) y* ***B*** *(mayor)*

c. Diagramación

Sln. Diagrama libre

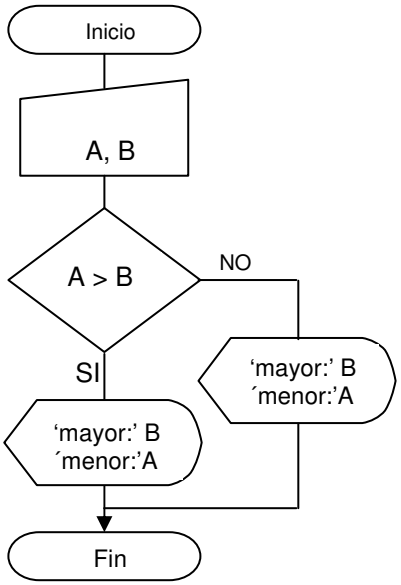
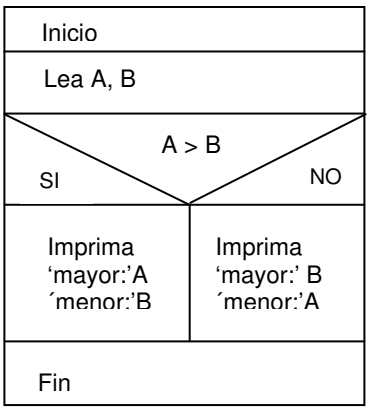


Diagrama estructurado

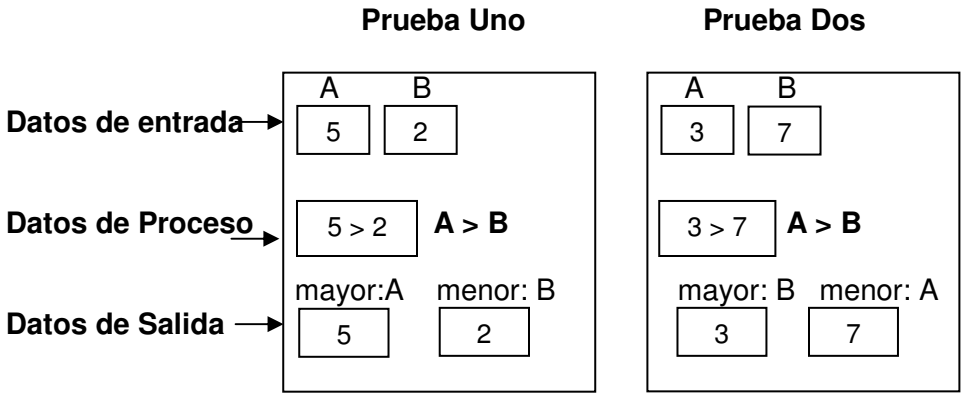


d. Seudocódigo

```
Inicio
Lea A, B
Si  (A > B) entonces
    Imprima ´mayor:' A, ´menor:´ B
Sino
    Imprima ´mayor:' B, ´menor:´ A
FinSi
Fin
```

e. Prueba de escritorio

Nuevamente se analizara dos pruebas de escritorio:



7. EJEMPLO 2

a. Definición

Dados tres números A, B y C, determinar si la suma de una pareja de ellos es igual al tercer número, si se cumple esta condición imprima un mensaje que diga “Iguales” y en caso contrario imprima “Distintos”.

b. Análisis

Datos de entrada

- *A, B, C*

Datos de Proceso

- *A = (B + C)* *ó*
- *B = (A + C)* *ó*
- *C = (A + B)*

Mensaje de salida

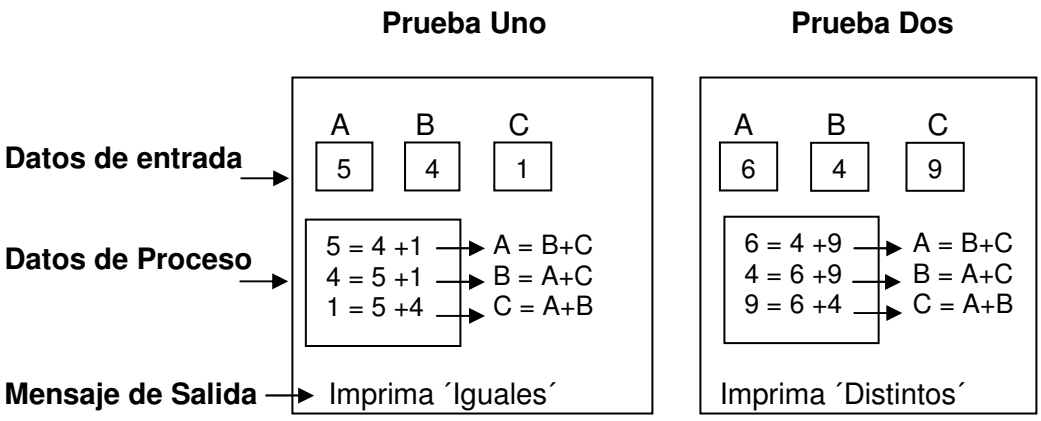
- *“Iguales” ó*
- *“Distintos”*

como pueden observar, no hay datos de salida únicamente se imprime el mensaje dependiendo si se cumple o no la condición.

c. Seudocódigo

```
Inicio
Lea A, B, C
Si  (A = B + C) entonces
      Imprima: ´Iguales´
Sino
     Si  (B = A + C) entonces
          Imprima: ´Iguales´
     Sino
          Si  (C = A + B) entonces
               Imprima: ´Iguales´
            Sino
                 Imprima: ´Distintos´
            FinSi
     FinSi
FinSi
Fin
```

d. Prueba de escritorio



8. EJEMPLO 3

a. Definición

Determinar si un número dado **N** es positivo, negativo o cero.

b. Análisis

Dato de entrada

- *N*

Datos de Proceso

- *N = 0* *para determinar si es cero*
- *N > 0* *para determinar si es positivo*
- *N < 0* *para determinar si es negativo*

Mensaje de salida

- *´El número es cero´* *ó*
- *´El número es positivo´ó*
- *´El número es negativo´*

c. Diagramación

SIn. ***Diagrama libre***

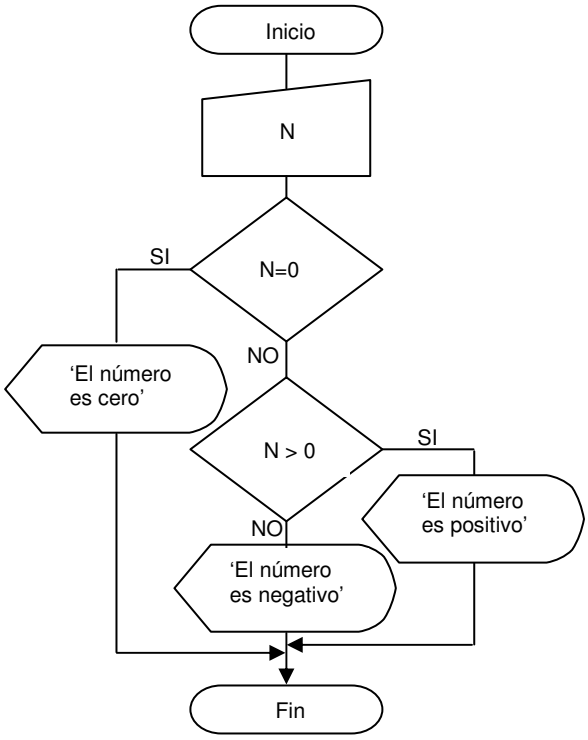
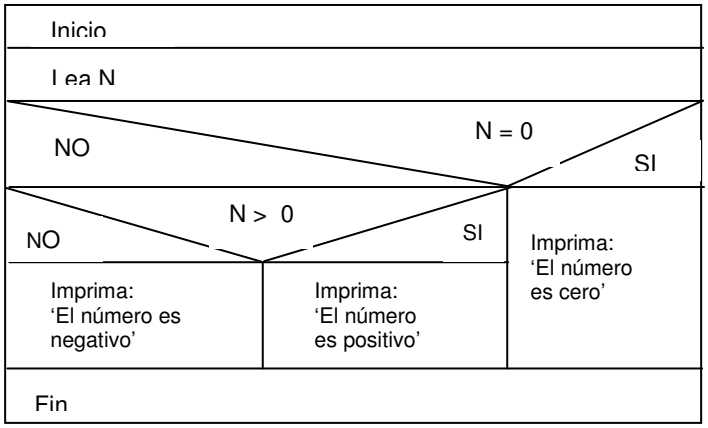


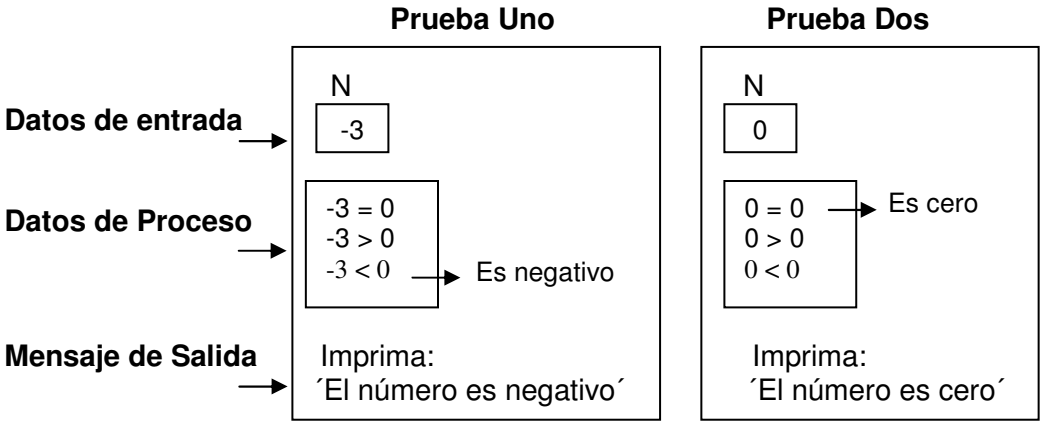
Diagrama estructurado



d. Seudocódigo

```
Inicio
Lea N
Si  (N = 0) entonces
      Imprima: ´El número es cero´
Sino
     Si  (N > 0) entonces
          Imprima: ´El número es positivo´
     Sino
           Imprima: ´El número es negativo´
     FinSi
FinSi
Fin
```

e. Prueba de escritorio



9. EJEMPLO 4

a. Definición

Diseñar un algoritmo para resolver una ecuación de segundo grado:

$$ax^2 + bx - c = 0$$

La ecuación de segundo grado tiene dos soluciones o raíces que son:

Sol. 1. $x_1 = \frac{-b + \sqrt{b^2 - 4ac}}{2a}$ Sol. 2. $x_2 = \frac{-b + \sqrt{b^2 - 4ac}}{2a}$

Pero antes de resolver la ecuación de segundo grado hay que tener en cuenta lo siguiente:

Discriminante

1°. Si $(b^2 - 4ac) < 0$: La ecuación no tiene solución

2°. Si $(b^2 - 4ac) = 0$: $x_1 = x_2$

Por lo tanto $x_1 = \frac{-b}{2a}$ y $x_2 = \frac{-b}{2a}$

3°. Si $(a = 0)$: La ecuación no tiene solución

b. Análisis

Datos de entrada

- *a, b y c*

Datos de Proceso

- Si $(b^2 - 4ac) < 0$ *imprimir mensaje "La ecuación no tiene solución"*
- Si $(a = 0)$ *imprimir mensaje "La ecuación no tiene solución"*
- Si $(b^2 - 4ac) = 0$ calcular $x_1 = \frac{-b}{2a}$ y $x_2 = \frac{-b}{2a}$

Caso de no cumplirse las condiciones anteriores calcular las soluciones:

- $x_1 = \frac{-b + \sqrt{b^2 - 4ac}}{2a}$ calcular solución uno

- $x_2 = \frac{-b + \sqrt{b^2 - 4ac}}{2a}$ calcular solución dos

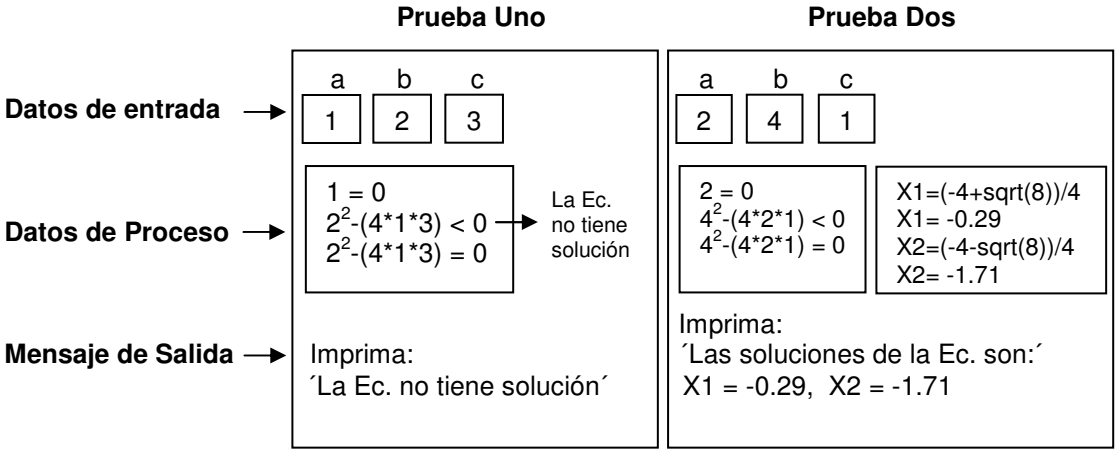
Datos de salida

- *Si existe solución:* x_1, x_2 ,
- *Si no: se imprime el mensaje ´La ecuación no tiene solución´*

c. Seudocódigo

```
Inicio
Lea a, b, c
discr = b^2 – (4*a*c)
Si  (a = 0) entonces
      Imprima: ´La Ec. no tiene solución´
Sino
     Si  (discr < 0) entonces
         Imprima: ´La Ec. no tiene solución´
      Sino
          Si  (discr = 0) entonces
               X1 = - b / (2 * a)
               X2 = - b / (2 * a)
          Sino
               X1 = (- b +SQRT(discr))/(2*a)
               X2 = (- b -SQRT(discr))/(2*a)
          FinSi
          Imprima: ´Las soluciones de la Ec. son:´ X1, X2
     FinSi
FinSi
Fin
```

d. Prueba de escritorio



10. EJEMPLO 5

a. Definición

Realizar un algoritmo para determinar si un número dado es par o impar.

b. Análisis

Dato de entrada

- *num* — *número que nos suministran*

Dato de Proceso

- *Si (num mod 2 = 0)* — *es par, si el residuo de dividir el número por dos (2) es cero, de lo contrario es impar.*

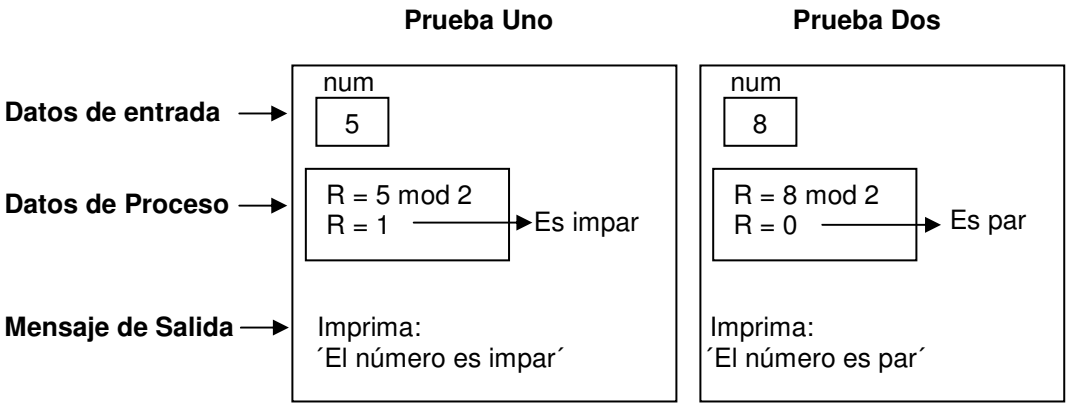
Mensaje de salida

- *´El número es par´* *ó*
- *´El número es impar´*

c. Seudocódigo

```
Inicio
Lea num
R = (num mod 2)
Si  (R = 0) entonces
      Imprima: ´El número es par´
Sino
    Imprima: ´El número es impar´
FinSi
Fin
```

d. Prueba de escritorio



11. EJEMPLO 6

a. Definición

Dado tres números, realizar un algoritmo para determinar el menor, el medio y el mayor.

b. Análisis

Datos de entrada

- *a, b, c*

Datos de Proceso

- *Si (a > b) y (a > c) ó para determinar si un número es mayor que otro hay que hacer comparaciones sucesivas,*
- *Si (b > a) y (b > c) ó después de determinar cual es el mayor, se debe establecer cual es el medio y el menor.*

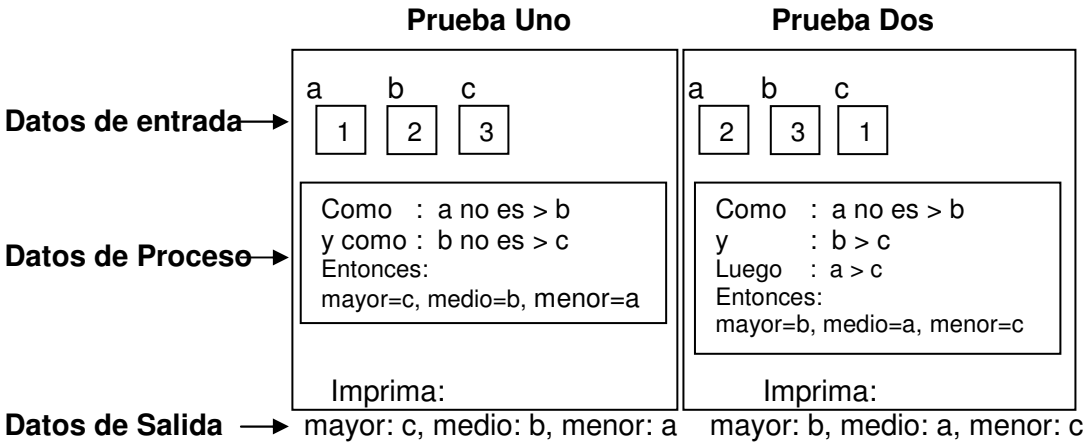
Datos de salida

- *a, b, y c indicando cual es mayor, medio y menor*

c. Seudocódigo

```
Inicio
Lea a,b,c
Si (a > b) entonces
    Si (a > c) entonces
        Si (b > c) entonces
            Imprima: ´mayor:´ a, ´medio:´ b, ´menor:´ c
        Sino
            Imprima: ´mayor:´ a, ´medio:´ c, ´menor:´ b
        FinSi
    Sino
        Imprima: ´mayor:´ c, ´medio:´ a, ´menor:´ b
    FinSi
 Sino
    Si (b > c) entonces
        Si (a > c) entonces
            Imprima: ´mayor:´ b, ´medio:´ a, ´menor:´ c
        Sino
            Imprima: ´mayor:´ b, ´medio:´ c, ´menor:´ a
        FinSi
    Sino
        Imprima: ´mayor:´ c, ´medio:´ b, ´menor:´ a
    FinSi
 FinSi
Fin
```

d. Prueba de escritorio



ACTIVIDADES

Diseñe un algoritmo para cada uno de los problemas planteados con su respectiva prueba de escritorio.

1. Leer tres valores por pantalla, determinar e imprimir el mayor, el medio y el menor.

2. Leer cuatro valores por pantalla, termine e imprima el mayor de todos.

3. Se desea obtener la nómina mensual de un empleado (salario neto) cuyo trabajo se paga por horas de la siguiente forma:

- Las horas menores o iguales a 120 (normales) se pagan a una valor determinado que es ingresada por teclado al igual que el número de horas y el nombre del trabajador.
-
- Las horas por encima de 120 se pagarán como extras al doble de las normales.
- La deducción de impuestos se hace con base al salario mensual, es decir, el valor de las horas mensuales normales + el valor de las horas extras.

  - Si el salario mensual es menor o igual a 380.000 pesos, se hace una deducción del 10%.
  - Si el salario mensual es mayor a 380.000 y menor o igual 480.000 se hace una deducción del 20%.
  - Si el salario mensual es mayor a 480.000 se hace una deducción del 30%.

4. Diseñe un algoritmo para verificar si un año que es ingresado por teclado es bisiesto o no, caso de ser cierto imprima un mensaje indicándolo.

5. Dado un número entre 1 y 30 determine si es primo o no, luego imprima un mensaje que indique tal caso.

6. Leer por pantalla un número entre 1 y 12 luego imprima en letras el mes a que corresponda dicho valor.

7. Diseñar un algoritmo para que lea una fecha de nacimiento de una persona, luego imprima lo siguiente:

- Número de años, meses, días.

8. Una distribuidora de huevos quiere contratarlo a usted, para que realice un algoritmo que calcule el precio de venta para una cantidad de huevos a llevar por un determinado cliente.

El precio de venta se calcula con base en la siguiente tabla:

Precio Unit. Huevo		Descuento por Cantidades a Llevar			
Tipo A	**Tipo AA**	**1-100**	**101-200**	**201-300**	**>= a 301**
$ 220.000	$ 250.000	3%	5%	8%	10%

8. Determinar el valor total a pagar por una llamada telefónica, de acuerdo a lo siguiente:

- Toda llamada que dure menos de tres minutos (<= 3) tiene un costo de $750 pesos.
- Cada minuto adicional-local a partir de los tres primeros tiene un costo de $200 pesos.
- Cada minuto adicional-nacional a partir de los tres primeros tiene un costo de $300 pesos.
- A demás por cada fracción de minuto se cobra la mitad de lo que vale un minuto local o nacional.

Nota: se debe leer por pantalla los minutos y segundos que duró la llamada.

9. Diseñar un algoritmo para que lea tres lados, determinar si forman un triangulo e indique el tipo de triangulo que forma: equilátero, isósceles, escaleno.

10. Dado un capital inicial **P** equivalente a **$800.000,** se desea encontrar el valor futuro **F** para las siguientes tasas de interés (***i_1=0.02 y i_2=0.08***) con periodos (***n_1= 5 y n_2=13***) respectivamente.

Tenga en cuenta que: $F = P(1 + i)^n$
Donde:

F = valor futuro
P = capital inicial
n = periodos
i = tasa de interés

11. Elaborar un algoritmo para que calcule la nota definitiva de un estudiante de lógica, se debe leer las siguientes notas por pantalla: Seg (seguimiento - 60%), parc1 (parcial uno -20%) y parc2 (parcial dos - 20%), al final debe imprimir un mensaje que indique si gano o perdió la materia.

12. Un almacén de venta de ropa tiene las siguientes promociones para sus clientes las cuales consisten en lo siguiente:

- Para ventas menores ó iguales a 100.000, se hace un descuento del 15%
- Para ventas mayores a 100000 y menores o iguales a 200.0000, se hace el 25%
- Para ventas mayores a 200.000, se hace un 35% de descuento

13. Un almacén de venta de ropa tiene las siguientes promociones para sus clientes las

cuales consisten en lo siguiente:

- Para ventas menores ó iguales a 100.000 con pago en efectivo, se hace un descuento del 20%, con tarjeta de crédito se hace el 10%
- Para ventas mayores a 100.000 y menores o iguales a 200.0000, con pago en efectivo se hace un descuento del 30%, con tarjeta de crédito se hace el 15%
- Para ventas mayores a 200.000, con pago en efectivo se hace un descuento del 40% y con tarjeta de crédito se hace el 20%.

ESTRATEGIAS METODOLÓGICAS

- ✓ Presentación de la unidad por parte del profesor.
- ✓ Elaboración y explicación magistral de los algoritmos por parte del docente.
- ✓ Elaboración de pruebas de escritorio por cada ejercicio desarrollado.
- ✓ Codificación de los algoritmos en el lenguaje acordado.

RECURSOS

- ✓ Humanos: profesor y alumnos.
- ✓ Institucionales: Salón de clases.
- ✓ Materiales: Módulo guía y tablero.

INDICADORES DE EVALUACIÓN

- ✓ Evaluación escrita sobre la unidad

CRITERIOS DE EVALUACIÓN

- ✓ Identificar las consecuencias en la toma de decisiones para resolver problemas por medio del computador.
- ✓ Actuar en la resolución de problemas utilizando estructuras de decisión.
- ✓ Encadenar y asociar la toma de decisiones para lograr un fin determinado.

UNIDAD 6

ESTRUCTURAS CÍCLICAS O REPETITIVAS

INTRODUCCIÓN

Otro paso en la elaboración de un algoritmo es el de la iteración de una o mas instrucciones involucradas en la solución de un requerimiento, por tanto existen dos esquemas el cualitativo y el cuantitativo en los cuales el programador o desarrollador debe tener agilidad y destreza para su selección; estos a su vez se componen de las siguientes estructuras (mientras, haga mientras que y para).

Es la herramienta de algoritmia que le permite simplificar y/o agrupar en su interior algún contador, condicional y acumulador que son temas previos, en la repetición de instrucciones en un momento y espacio determinado.

JUSTIFICACIÓN

La elaboración de algoritmos en su gran mayoría hace indispensable el empleo de repeticiones para evitar extender el código con las mismas actividades dentro de un proceso, estas a su vez tienen elementos que le facilitan dicho funcionamiento de manera implícita o explicita al igual que la forma de presentarla e interpretarla. El empleo de esta estructura permite un mejor desempeño del desarrollador resumiendo el manejo de las condicionales y sus operadores. Una solución cíclica dinamiza la operación a la vez, permite el rompimiento de controles.

OBJETIVO

Conocer la forma como funcionan las estructuras repetitivas, para dar solución a problemas en los cuales sea necesario repetir la ejecución de procesos o procedimientos un determinado numero de veces.

OBJETIVOS ESPECÍFICOS

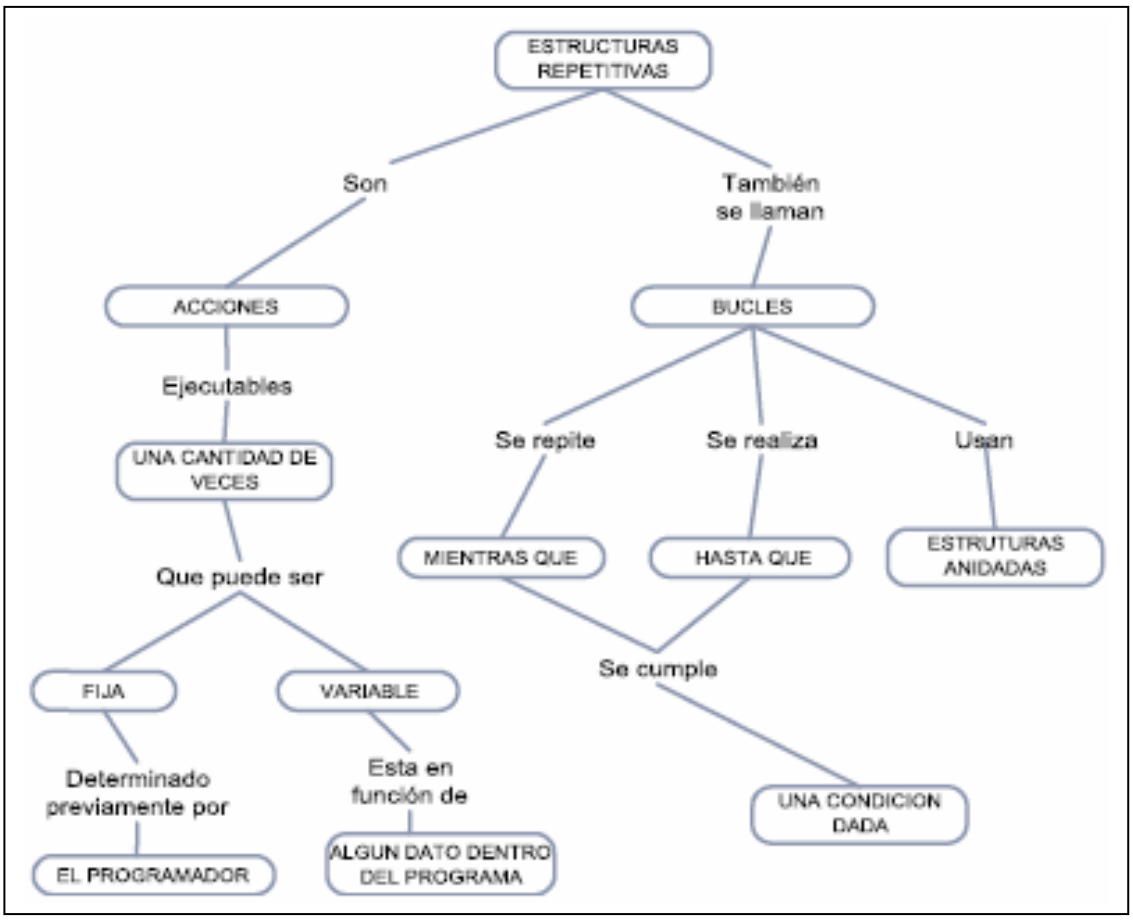
1. Identificar cada una de las estructuras cíclicas.
2. Clasificar las diferentes estructuras repetitivas.
3. Reconocer las estructuras cualitativas y cuantitativas empleadas al momento de realizar un algoritmo.
4. Desarrollo de algoritmos empleando cada una de las estructuras cíclicas.
5. Conocer las estructuras sintácticas para formar en el estudiante la disciplina. requerida para afrontar las soluciones de problemas en forma algorítmica.
6. Identificar las herramientas necesarias para utilizar el computador mediante un lenguaje de programación, como instrumento de trabajo en su vida académica.

CONTENIDO

1. Estructuras cíclicas o repetitivas.
2. Esquemas: cuantitativo y cualitativo.
3. Estructura Mientras.
4. Rompimientos de: ciclos y control de ejecución.
5. Estructuras Para
6. Estructura Hacer mientras que.

1. ESTRUCTURAS CÍCLICAS O REPETITIVAS

Una estructura cíclica permite agrupar una o más instrucciones (bloque). Esta o estas se repite un número determinado o no de veces de acuerdo con el enunciado a solucionar, a partir de este se definen dos esquemas: el cuantitativo y el cualitativo.



Mapa conceptual Estructuras Repetitivas[8]

2. ESQUEMAS: CUANTITATIVO Y CUALITATIVO

El esquema cualitativo se caracteriza desde el enunciado, al no determinar el número de repeticiones a efectuar, puede ser infinito por tanto se debe incluir un condicional (centinela, bandera o swiche), quien controlara dicho proceso.

Un esquema es cualitativo cuando la cantidad de iteraciones no se puede conocer antes de éste activarse y es el usuario quien determina cuando terminar de repetir las acciones o instrucciones del ciclo.
Un esquema es cuantitativo cuando la cantidad de iteraciones puede conocerse antes de éste activarse (porque previamente se lo especifican al programador o porque el usuario del programa le puede dar un valor)

El cuantitativo se caracteriza también porque desde el enunciado se determina el número de repeticiones a efectuar. Este es finito.

[8] Tomado del libro "Fundamentos de programación con énfasis en análisis y metodología para trabajo en equipos efectivos"

3. ESTRUCTURA MIENTRAS HACER

Se llaman problemas repetitivos o cíclicos a aquellos en cuya solución es necesario utilizar un mismo conjunto de acciones que se puedan ejecutar una cantidad específica de veces dependiendo del valor de verdad que resulte de evaluar una condición de tipo lógico.

Cada vez que se ejecutan las instrucciones de un ciclo se dice que se hizo una iteración. Por lo tanto, si el ciclo se repite 5 veces por ejemplo, se hicieron 5 iteraciones.

La forma general del ciclo MIENTRAS es:

```
MIENTRAS <Condición> HACER
        <Secuencia de instrucciones>
        <actualizacion>
FIN_MIENTRAS
```

Dentro de la secuencia de instrucciones debe haber por lo menos una que, en algún momento cambie el valor de verdad de la condición, pues de lo contrario se repetiría la ejecución de la secuencia en forma indefinida (ciclo infinito).

El funcionamiento es como sigue: Al llegar a la estructura MIENTRAS se evalúa la condición (expresión lógica); si el resultado de la evaluación es verdadero, se LEE el primer registro de datos y se procesa y a continuación se vuelve a evaluar la condición y si nuevamente es verdadera, se lee el siguiente registro y se procesa. Si el resultado de la evaluación de la condición es falso, se continúa con la siguiente instrucción que haya después del FINMIENTRAS (si la hay).
Es posible que la primera vez que se evalúa la condición, el resultado de su evaluación sea falso. En este caso no se ejecuta nunca las instrucciones dentro del ciclo. Es por esto que el ciclo MIENTRAS se dice que ejecuta cero o más veces una secuencia de instrucciones.

La representación general de este esquema es:

```
INICIO
        <Inicializar El contador>
        LEER <Nº de iteraciones>
        MIENTRAS <Condición> HACER
                LEER <Registro de datos>
                <PROCESO>
        FIN_MIENTRAS
        .
FIN_INICIO
```

En este tipo de esquema, se controla la repetición del ciclo con una variable tipo contador, la cual se compara con el número de iteraciones que debe hacer el algoritmo o con el número hasta el cual debe llegar. Dentro del PROCESO debe haber una instrucción de incrementar o disminuir el valor del contador. Esta instrucción es la que hace que la condición se haga falsa en determinado momento y que el ciclo termine (no sea infinito).

Ejemplo:

Hacer un algoritmo que sume los cinco primeros números naturales.

Análisis del problema:

Datos de entrada: No hay, ya que los número naturales (1, 2, 3, etc) no se ingresan por parte del usuario sino que se deben generar por el algoritmo (no hay una instrucción de lectura).
Datos de salida: La suma de esos primeros cinco números naturales.

Definición de variables:

NUM: Variable para guardar el número natural generado.
SUMA: Variable para guardar la suma de los números naturales.

Algoritmo:

```
INICIO
        NUM=0
        SUMA=0
        MIENTRAS NUM<5 HACER
                NUM=NUM+1
                SUMA=SUMA+NUM
        FIN_MIENTRAS
        IMPRIMIR "La suma de los cinco primeros números naturales es ", SUMA
FIN_INICIO
```

Otra forma de resolver el problema es:

Algoritmo

```
INICIO
        NUM=1
        SUMA=0
        MIENTRAS NUM<=5 HACER
                SUMA=SUMA+NUM
                NUM=NUM+1
        FIN_MIENTRAS
        IMPRIMIR "La suma de los cinco primeros números naturales es ", SUMA
FIN_INICIO
```

Esquema cualitativo

La representación general de este esquema es:

```
INICIO
        LER <Campo elegido>
        MIENTRAS <Condición> HACER
                LEER <Resto del registro de datos>
                <PROCESO>
                LEER < Campo elegido >
        FIN_MIENTRAS
        .
        .
FIN_INICIO
```

En este tipo de esquema, se controla la repetición del ciclo con el valor del campo elegido, al cual el usuario, en el momento que desee terminar la repetición del ciclo, le asigna un valor que no esté contemplado en el rango de valores que este campo pueda tomar. Esta instrucción de lectura (entrada de datos) es la que hace que la condición se haga falsa en determinado momento y que el ciclo termine (no sea infinito). El valor no válido para el campo elegido se guarda en un registro llamado "registro centinela" o "registro marca" y tiene el significado de "no hay más registros para procesar".

Ejemplo:

Hacer un algoritmo que encuentre la suma y el promedio de un grupo de números positivos que entran de a uno por registro.

Análisis del problema:

Datos de entrada: Los números que el usuario ingresa de a uno.
Datos de salida: La suma de los números y el promedio de los números
Proceso: El algoritmo le dirá al usuario que le ingrese cada número. Este le ingresará número positivos hasta que desee terminar, caso en el cual le ingresará un 0 o un número negativo. Lo más recomendable es que el algoritmo le diga al usuario con qué valor terminar. Los números ingresados se deben ir contando en una variable tipo contador y se deben acumular sus valores en una variable tipo acumulador.

Definición de variables:

NUM: Variable para guardar el número entrado.
SUMA: Variable para guardar la suma de los números.
PROM: promedio de los números
CN: contador de números ingresados.

Algoritmo:

```
INICIO
        Imprimir “Ingrese un número positivo “
        LEER NUM
        SUMA=0
        CN=0
        MIENTRAS NUM <> 0 HACER
                SUMA=SUMA+NUM
                CN= CN+1
                Imprimir “”Ingrese otro número positivo o 0 para terminar “
                LEER NUM
        FIN_MIENTRAS
        PROM = SUMA/CN
        IMPRIMIR “La suma es “,SUMA, “ Y EL PROMEDIO ES “,prom
FIN_INICIO
```

Como se puede apreciar, si el usuario desea estar indefinidamente ingresando números positivos, lo puede hacer y el algoritmo le sigue pidiendo más y más números, pero si él desea terminar de ejecutarlo, le ingresa un 0 y con eso la condición se hace falsa y ya no se ejecutará más la secuencia de instrucciones dentro del ciclo. El registro que contiene dicho 0 es el que se denomina “centinela”.

Problemas (Mientras o Mientas que)

Nota: Se deja a los alumnos el trabajo de hacer el análisis del problema, o sea escribir cuáles son los datos de entrada, los de salida y hacer la definición de variables de acuerdo al planteamiento del problema y del algoritmo. Igualmente, complementarlo con instrucciones de salida de forma que sea más amigable con el usuario. Por ejemplo, antes de la instrucción Leer Nom, se le puede añadir la instrucción Imprimir “Ingrese el nombre del estudiante: “

1) Calcular el promedio de un alumno que tiene 7 calificaciones en la materia Física.(Esquema cuantitativo)

```
Inicio
  Sum=0
  CC=0
  Leer Nom
  MIENTRAS CC < 7 HACER
        Leer calif
        Sum = sum + calif
        CC=CC+1
  Fin-Mientras
  prom = sum /7
  Imprimir prom
Fin.
```

2) Leer una cantidad indeterminada de números diferentes de cero y obtener su cubo y su cuarta. (Esquema cualiitativo)

```
Inicio
Leer num
  MIENTRAS CONT<>0 HACER
        cubo = num * num * num
        cuarta = cubo * num
        Imprimir cubo, cuarta
        Leer num
  FIN_MIENTRAS
Fin.
```

3) Leer N números e imprimir solamente los números positivos. (Esquema cuantitativo)

```
Inicio
  CN = 0
  Leer N
  MIENTRAS CN<N HACER
        Leer num
        Si num > 0 entonces
              Imprimir num
        fin-si
        CN = CN + 1
  FIN_MIENTRAS
Fin.
```

4) Leer N números e imprimir cuántos son positivos, cuántos negativos y cuántos ceros. (Esquema cuantitativo)

```
Inicio
  cneg = 0
  cpos = 0
  cceros = 0
  MIENTRAS CONT < N HACER
        Leer num
        Sin num = 0 entonces
              cceros = cceros + 1
        si no
              Si num > 0 entonces
                    cpos = cpos + 1
              si no
                    cneg = cneg + 1
              Fin-si
        Fin-si
        CONT 0 CONT + 1
  FIN_MIENTRAS
  Imprimir cceros, cpos, cneg
Fin.
```

5) Leer 15 números negativos y convertirlos a positivos e imprimir dichos números.

```
Inicio
  MIENTRAS X<15 HACER
        Leer num
        pos = num * (-1)
        Imprimir num, pos
              X = X + 1
  FIN_MIENTRAS
Fin.
```

6) Suponga que se tiene la nota de una materia de un grupo de 40 alumnos. Realizar un algoritmo para calcular la nota promedio y la nota más baja de todo el grupo.

```
Inicio
  CE = 0
  sum = 0
  baja = 9999
  MIENTRAS CE<40 HACER
        Leer calif
        sum = sum + calif
        Si calif < baja entonces
              baja = calif
        fin-si
        CE=CE+1
  FIN_MIENTRAS
  media = sum / 40
  Imprimir media, baja
fin
```

7) Calcular e imprimir la tabla de multiplicar de un número cualquiera de 1 hasta 10. Imprimir el multiplicando, el multiplicador y el producto.

```
Inicio
  X=1
  Leer num
  MIENTRAS X<=3 HACER
        resul = num * X
        Imprimir num, " * ", X, " = ", resul
        X=X+1
  FIN_MIENTRAS
fin.
```

8) Simular el comportamiento de un reloj digital, imprimiendo la hora, minutos y segundos de un día desde las 0:00:00 horas hasta las 23:59:59 horas:

```
Inicio
  h=0
  MIENTRAS h<24 HACER
    m=0
    MIENTRAS m<60 HACER
        s=0
        MIENTRAS s<60 HACER
            Imprimir  h, m, s
            s=s+1
        FIN_MIENTRAS
        m=m+1
    FIN_MIENTRAS
    h=h+1
  FIN_MIENTRAS
fin.
```

4. Rompimiento De Cíclos Y Control De Ejecución

En la programación, a veces es necesario que se aborte o suspenda un ciclo para que no continúe repitiendo sus instrucciones, debido a que ya se cumplió algo que se estaba buscando y es posible que esto se haga sin haberse completado el número de iteraciones que se tenía programado ejecutar. Esto se conoce como ruptura de ciclos y se consigue cambiando el valor de verdad de la expresión lógica de verdadero a falso.

Ejemplo del caso (b):

Se tiene el nombre y la edad de N personas y se desea saber si dentro de ellas hay por lo menos una que sea menor de edad y el nombre de la primera que se encuentre.

Análisis del problema:

Datos de entrada: El nombre y la edad de la persona
Datos de salida: Un mensaje que diga si se encontró un menor de edad y su nombre.

Definición de variables:

NP: Número de personas.
NOM: Variable para guardar el nombre.
EDAD: Edad de la persona.
CP: Contador de personas.
ENCONTRADO: Variable tipo bandera que permitirá ROMPER el ciclo si se encuentra un menor de edad.

Proceso: Como se puede observar, el esquema es cualitativo y se puede controlar comparando el contador de personas con el número de personas a procesar, a la condición se le adicionará el switche para que, si se encuentra un menor de edad, ésta se haga falsa y se rompa el ciclo aunque no se llegue a procesar todas las N personas. Si se termina de procesarlas a todas y no se encuentra una menor de edad, el ciclo también se termina, pero ya porque la otra condición (CP<NP) es falsa..

Algoritmo.

```
INICIO
        CO=0
        ENCONTRADO=.F.
        Leer NP
        MIENTRAS (CP<NP) AND (ENCONTRADO=.F.) HACER
                Leer NOM, EDAD
                Si EDAD<18 entonces
                        ENCONTRADO=.V.
                Sino
                        CP=CP+1
                Fin_si
        FIN_MIENTRAS
        Si ENCONTRADO=.V. Entonces
                Imprimir NOM
        Sino
                Imprimir “No hay ningún menor de edad en el grupo”
        Fin_si
FIN_INICIO
```

5. ESTRUCTURA PARA

Permite ejecutar una o varias instrucciones mientras los valores de una progresión aritmética de razón creciente o decreciente se vayan asignando a una variable llamada “variable de control del ciclo para”. El control del ciclo se hace en forma automática, con base en parámetros que establece el programador.

Esta estructura se puede usar en reemplazo del ciclo Mientras en esquema cuantitativo cuando el contador que controla dicho ciclo se incrementa o disminuye en un valor constante. La diferencia con el ciclo Mientras es que en el ciclo PARA, la variable controladora se inicializa, se incrementa y se compara automáticamente.

Representación o forma general

```
PARA VC = LI, LF, INC   HACER
        Accion1
        Accion2
           .
FIN_PARA
```

Donde:

VC	Variable de control del ciclo
LI	Limite inicial
LF	Limite final
INC	Incremento

En este ciclo la variable de control toma el valor inicial del ciclo y el ciclo se repite hasta que la variable de control llegue al límite final.

Funcionamiento:

a. Si INC es positivo: (LI < LF)

Cuando se activa la estructura, automáticamente se asigna el valor de LI a VC y se compara VC con LF. Si VC>LF, no se ejecuta la secuencia de instrucciones o acciones. Si VC<=LF, se ejecuta la secuencia una vez y automáticamente regresa al principio del ciclo a incrementar la variable controladora en el valor de INC (VC=VC+INC) y se compara nuevamente el valor de VC con LF. Si nuevamente VC<=LF, se vuelve a ejecutar la secuencia y se regresa al principio del ciclo a repetir lo mismo. Si ahora, VC>LF, se termina el ciclo automáticamente.

b. Si INC es negativo:

Se hace lo mismo pero se ejecuta la secuencia siempre y cuando VC>=LF y no se ejecuta si VC<LF. En ese caso LI>LF

Ejemplo:

Hacer un algoritmo que encuentre el factorial de un número positivo cualquiera.

Análisis del problema:

Datos de entrada: El número al cual se le hallará el factorial.
Datos de salida: El factorial del número.
Proceso: El factorial de un número es el producto del número por todos sus números anteriores hasta uno, así: Factorial de 5= 5*4*3*2*1. Se debe entonces inicializar el factorial en 1 (módulo del producto) incrementar un contador y multiplicar por él (esto lo hace el ciclo PARA)

Definición de variables:
N: Número al cual se le hallará el factorial
FAC: Factorial del número
CON: contador que generará los factores del factorial.

Algoritmo:

```
INICIO
        Leer N
        FAC=1
        PARA CON=1, N, 1 HACER
                FAC=FAC*CON
        FIN_PARA
        Imprimir "El factorial de ",N, " es: ", FAC
FIN_INICIO
```

Ejemplo:

Hacer un algoritmo que imprima las tablas de multiplicar de 1 a 5.

Definición de variables:

I: multiplicando
J: Multiplicador
P: Producto

Algoritmo:

```
INICIO
        PARA I=1,5,1 HACER
                PARA J=1,5,1 HACER
                        P=I*J
                        Imprimir I, "*", J, "=", P
                FIN_PARA
        FIN_PARA
FIN_INICIO
```

Ejemplo:

Hacer un algoritmo que encuentre la suma de los números impares comprendidos entre 1 y N.

Definición de variables:

N: Número hasta el cual se generarán números impares
IMP: Número impar generado (el algoritmo los genera, no se leen o ingresan por el usuario)

SUMA: Suma de los números impares.

Algoritmo:

```
INICIO
        Leer N
        SUMA=0
        PARA IMP=1,N,2 HACER
                SUMA=SUMA+IMP
        FIN_PARA
        Imprimir "La suma de los impares comprendidos entre 1 y ", N, " es: ", SUMA
FIN_INICIO
```

6. ESTRUCTURA REPETIR (HACER MIENTRAS QUE)

Esta es una estructura similar en algunas características, a la estructura MIENTRAS. Repite un proceso una o varias veces, a diferencia del Mientras, el cual lo repite cero o más veces, esto debido a que el REPETIR evalúa la condición al final del proceso, mientras que en el Mientras puede ser que nunca llegue a entrar si la condición no se cumple desde un principio.

La forma de esta estructura es la siguiente:

```
REPETIR
        Acción 1
        Acción 2
          .
          .
        Acción N
MIENTRAS <condición>
```

Funcionamiento:

Al entrar a la estructura se ejecuta la secuencia de acciones o instrucciones una vez y se evalúa la condición. Si ésta es falsa, se sale del ciclo y se continúa con la siguiente instrucción; si es verdadera, se ejecuta nuevamente la secuencia y se vuelve a evaluar la condición. El proceso se repite mientras la condición sea verdadera.

Ejemplo:

Hacer un algoritmo que encuentre la suma de los primeros N números naturales.

Algoritmo:

```
INICIO
        SUMA=0
        NUM=1
        Leer N
        REPETIR
                SUMA=SUMA+NUM
                NUM=NUM+1
        MIENTRAS NUM<=N
        Imprimir "La suma de los primeros ", N, " números naturales es: ", SUMA
FIN_INICIO
```

ACTIVIDADES

1) Responder al siguiente cuestionario

1. Las formas de representar esquemáticamente un proceso cíclico son:

a. *Reflexivo y el cuantitativo*
b. *cualitativo y el cuantitativo.*
c. *cualitativo y el reflexivo*

2. La herramienta en algoritmia que le permite simplificar y /o agrupar en su interior algún contador, algún condicional y algún acumulador se conoce como

a. *Preguntas.*
b. *Ciclos.*
c. *Procesos.*

3. Como define el esquema cualitativo

a. Desde el enunciado se caracteriza la no determinación del número de repeticiones a efectuar.
b. Desde el enunciado se caracteriza la determinación del número de repeticiones a efectuar.
c. Desde el enunciado se caracteriza la no determinación del número de preguntas a efectuar.

4. La siguiente definición El control del ciclo se hace en forma automática, con base en parámetros que establece el programador permite definir :

a. *Un ciclo para.*
b. *Un ciclo mientras.*

c. *Ninguno de los dos.*

5. ¿Como se diferencia un ciclo mientras de un ciclo repita hasta que?

a. Repite desde cero o más veces
b. No repite desde cero o más veces
c. Ninguna de las anteriores

ESTRATEGIAS METODOLÓGICAS

- ✓ Presentación de la unidad a cargo del profesor.
- ✓ La parte algorítmica y del lenguaje serán tratadas en forma simultánea
- ✓ El curso se abordará mediante clases que estimulen al estudiante a la creación de algoritmos y el desarrollo de su ingenio.
- ✓ Los algoritmos solucionados, serán implementados por computador mediante el lenguaje de programación C++.
- ✓ Para afianzar los conocimientos del curso, se harán como mínimo tres prácticas, independientes de los problemas solucionados en clase, donde se apliquen los conocimientos adquiridos.
- ✓ Realización por parte de los alumnos de las actividades propuestas en la unidad.

RECURSOS

- ✓ Humanos: Profesor y alumnos.
- ✓ Institucionales: Salón de clase.
- ✓ Materiales: Texto guía.

INDICADORES DE EVALUACIÓN

- ✓ Evaluación escrita sobre la unidad.
- ✓ Evaluación de las actividades propuestas en la unidad.
- ✓ Se realizarán evaluaciones teniendo en cuenta el calendario de entrega de informes. de las Instituciones Educativas (bimestrales).
- ✓ Bimestre 25%. Desde estructura mientras, hasta Estructuras adicionales.

CRITERIOS DE EVALUACIÓN

- ✓ ¿Cuales son los esquemas el en los cuales el programador o desarrollador debe tener agilidad y destreza?
- ✓ ¿Como se define el enunciado que no determina el numero de repeticiones?
- ✓ Cuáles son los tipos de ciclos?
- ✓ ¿En la programación, a veces es necesario que se aborte o suspenda un ciclo para que no continúe repitiendo sus instrucciones?,
- ✓ El control del ciclo se hace en forma automática, con base en parámetros?

BIBLIOGRAFÍA

- Algoritmos, conceptos básicos. Becerra, Cesar.
- Algoritmos. Peralta, Luis A.
- Desarrollo de algoritmos y sus aplicaciones. Correa Guillermo.
- Diagramación y programación. Lozano. Letvin.
- Introducción a la ciencia de los computadores. Tremblay. J. P.
- Fundamentos de programación. Joyanes, Luis.
- Fundamentos de programación con énfasis en análisis y metodología para trabajo en equipos efectivos. González, Luis F.
- Lógica de programación. Oviedo R, Efraín.
- Lógica para programación de computadores. Vasquez, Gabriel.
- Problemas de la metodología en la programación. Joyanes, Luis.
- Soluciones secuenciales. Ríos C, Fabián.