**INFORMATICA**

ACTIVIDAD ALGORITMO

2.

A. Existen otras propiedades que alcanzan a todos los algoritmos, con excepción de los denominados algoritmos paralelos: El **TIEMPO SECUENCIAL** (estos algoritmos funcionan paso a paso), El **ESTADO ABSTRACTO** (cada uno de los algoritmo son independiente de su implementación) y la **EXPLORACIÓN ACOTADA** (transición entre estados que queda determinada por la descripción finita y fija).

Y mencionando por último que los algoritmos son importantes en la [**INFORMÁTICA**](http://definicion.de/informatica/) y nos permiten representar datos como secuencias de bits. Un programa es un algoritmo y le indica a la [**COMPUTADORA**](http://definicion.de/computadora/) los pasos específicos a debe seguir para desarrollar múltiples tareas.

 B. **CARACTERÍSTICAS DE LOS ALGORITMOS**

Finitos: Termina en un momento determinado.

Eficientes: ocupan el espacio mínimo en la memoria, y hace minimizar el tiempo en alguna ejecución.

 Legibles: Tiene que ser un texto legible y comprensible, claro que sea fácil de entender y leer.

Modificables: Están diseñados de tal modo que posteriormente sea modificado y sean fáciles de hacer, y incluso que sean programados diferente a sus propios creadores.

Modulares: La filosofía utilizada para los diseños favorecen la división de los problemas en módulos pequeños.

Único punto de entrada, único punto de salida: A los algoritmos y a los módulos los integran varios puntos y entra por un sólo, inicio, y sale por un sólo punto también.

C.

 SEUDOCÓDIGO: Es una forma de diagramar o representar un algoritmo para resolver un determinado problema, independiente de cualquier lenguaje de programación en especial.

DIAGRAMA DE FLUJO: Es la esquematización gráfica de un algoritmo, el cual muestra gráficamente los pasos o procesos a seguir para alcanzar la solución de un problema. Es la representación de la solución utilizando figuras geométricas como circunferencias, flechas, rectángulos, rombos, etc.

DIAGRAMA N - S: También conocido como diagrama de Chapín es una técnica de especificación de algoritmos que combina la descripción textual, propia del seudocódigo, con la representación gráfica del diagrama de flujo.

EL DIAGRAMA N-S: Cuenta con un conjunto limitado de símbolos para representar los pasos del algoritmo, por ello se apoya en expresiones del lenguaje natural; sin embargo, dado que el lenguaje natural es muy extenso y se presta para la ambigüedad, solo se utiliza un conjunto de palabras

DIAGRAMA CHAPÍN :

EJEMPLO

* Aflojar tornillos de las llantas
* levantar el coche con el gato
* Sacar los tornillos de las llantas
* Quitar la llanta
* Poner la llanta de repuesto
* Poner los tornillos
* Bajar el gato
* Apretar los tornillos

**Fin**

D.

Es un gráfico lógico del plan de trabajo que se ejecutara para la solución de un determinado problema. A través de él, se planifica la solución del problema independiente del lenguaje de computación a usar. De esta manera se separa loas instrucción es un lenguaje determinado con todas las reglas.

Las capacidades humanas necesarias para elaborar un diagrama de flujo correcto son: Lógico, Prácticas, y Atención.

El empleo de la maquina en las funciones del procediendo de datos han hecho necesario un flujo ordenado de la información. La secuencia en que deberán ejecutarse las operaciones tendrá que definirse claramente, y cuando se combine con los datos a los que debe aplicarse, esa secuencia creara el flujo de información.

No puede hacerse mucho hincapié en documentación, ósea el registro de Información .Sin Instrucciones escritas y sin representación gráfica del flujo de trabajo sería muy difícil de llevar una tarea de procediendo de datos en forma apropiada. Hay varios métodos más eficientes organizados y normalizados, es el de los diagramas de Flujo que el Futuro programador comprenda la necesidad de los diagrama de flujo.

**Objetivos de un diagrama de flujo:**

Estructura la solución del problema independiente del lenguaje a utilizar.

Separar la solución lógica de programación de la parte de reglas y sintaxis de codificación con esta división del trabajo se obtiene mayor eficiencia.

Dar una visión completa del problema al programador ya que pierde en un programa ya codificado.

Permitir una compresión más rápida del programa a otros programadores.

**Tipos de diagramas de flujo:**

Diagrama de flujo de sistemas: muestra en que forma se procesan los datos, entre as principales funciones o estaciones de trabajo .En este diagrama completo de computadora se presenta con un solo símbolo de procesamiento.

**Ejemplo de Diagrama de Flujo de sistema:**



**Diagrama de flujos de programación:**
Son las operaciones y decisiones en la secuencia en que las ejecutará una computadora de procesamiento de datos. Los símbolos representan esas operaciones e indican el orden en que se ejecutaran. Por lo tanto, un diagrama de flujo de programa proporciona una descripción gráfica del programa.

**Ejemplo de Diagrama e Flujo de Programa:**



**Simbología de los diagramas de flujo:**

Las diversas organizaciones usan distintos símbolos, pero el comité sobre computadoras y procesadores de información de la Asociación Norteamericana de Normas ha hecho un gran esfuerzo para normalizar los símbolos de los diagramas de flujo. Esa normalización permite comprender cualquier diagrama de flujo que use los símbolos recomendados.

***Cada símbolo normal de diagrama de flujo tiene un significado especial.***

******

***Expresa Inicio o Fin de un Programa***.



***Expresa operación algebraica o de Asignación.***



***Expresa condiciones y asociaciones alternativas de una decisión***[***lógica***](http://www.monografias.com/trabajos15/logica-metodologia/logica-metodologia.shtml)***.***



***Expresa condición y acciones alternativas de una decisión numérica.***



***Entrada / Salida: Representa cualquier tipo de Fuente de entrada y salida***



***Entrada: Lectura de datos por tarjeta perforadas.***



***Conector dentro de página.***



***Representa resultado mediante un reporte impreso***



***Conector fuera de página.***



***Expresa operación cíclica repetitiva.***



***Expresa proceso de llamada a una subalterna.***



***Representa datos grabados en una cinta magnética***.



***Almacenamiento en línea Disco Magnético.***

E,

TIPOS DE ALGORITMOS Y PROBLEMAS.

**Cualitativos**: Son aquellos en los que se describen los pasos utilizando palabras.

* Son todos aquellos pasos o instrucciones descritos por medio de palabras que sirven para llegar a la obtención de una respuesta o solución de un problema cualquier

**Cuantitativos:** Son aquellos en los que se utilizan cálculos numéricos para definir los pasos del proceso.

* Son aquellos pasos o instrucciones que involucran cálculos numéricos para llegar a un resultado satisfactorio

**Tipos de algoritmos de razonamiento:**

**Algoritmos Estáticos:** son los que funcionan siempre igual, independientemente del tipo de problema tratado.

**Algoritmos Adaptativos**: algoritmos con cierta capacidad de aprendizaje.

**Algoritmos Probabilísticos:** son algoritmos que no utilizan valores de verdad booleanos sino continuos. Existen varios tipos de algoritmos probabilísticos dependiendo de su funcionamiento, pudiéndose distinguir.

***PROBLEMA:*** Realizar la suma de los números 2448 y 5746.

**ALGORITMO:**

|  |
| --- |
| **Inicio****PASO 1.** Colocar los números el primero encima del segundo, de tal manera que las unidades, decenas, centenas, etc., de los números queden alineadas. Trazar una línea debajo del segundo número.**PASO 2.**  Empezar por la columna más a la derecha.**PASO 3.**  Sumar los dígitos de dicha columna.**PASO 4.** Si la suma es mayor a 9 anotar un 1 encima de la siguiente columna a la izquierda y anotar debajo de la línea las unidades de la suma. Si no es mayor anotar la suma debajo de la línea.**PASO 5.**  Si hay más columnas a la izquierda, pasar a la siguiente columna a la izquierda y volver a 3.**PASO 6.** El número debajo de la línea es la solución.**Fin** |

**PROBLEMA:** Sean los puntos *P=(a,b)* y *Q=(c,d)* que definen una recta, encontrar un segmento de recta perpendicular a la anterior que pasa por el punto medio de los puntos dados.

**ALGORITMO:**

|  |
| --- |
| **Inicio**http://www.virtual.unal.edu.co/cursos/ingenieria/2001839/modulo1/cap_02/imagenes_02/image002.gif**PASO 1.** Trazar un círculo con centro en el punto P que pase por el punto Q.http://www.virtual.unal.edu.co/cursos/ingenieria/2001839/modulo1/cap_02/imagenes_02/image004.gif**PASO 2.** Trazar un círculo con centro en el punto Q que pase por el punto P.http://www.virtual.unal.edu.co/cursos/ingenieria/2001839/modulo1/cap_02/imagenes_02/image006.gif**PASO 3.** Trazar un segmento de recta entre los puntos de intersección de las circunferencias trazadas.http://www.virtual.unal.edu.co/cursos/ingenieria/2001839/modulo1/cap_02/imagenes_02/image008.gif**Fin.** El segmento de recta trazada es el buscado.http://www.virtual.unal.edu.co/cursos/ingenieria/2001839/modulo1/cap_02/imagenes_02/image010.gif |

F. Es una variable es un espacio de memoria reservado para el almacenamiento de valores y corresponda al tipo de datos que son soportados por el lenguaje de programación.

Las variables son representadas y usadas a través de etiquetas que define su creador.

F.1En la computación las variables pueden ser utilizadas en procesos repetitivos: se le puede asignar un valor en cualquier sitio, y ser luego utilizada en otro, más adelante reasignársele nuevos valores para más tarde utilizarlos de otra manera.

G. QUE SON LAS CONSTANTES ?

Una constante son datos cuyos valores no pueden cambiar durante la ejecución del programa. Estos reciben valores en el momento de la compilación y este permanece inalterado durante todo el programa. Las constantes se declaran en secciones que comienza con palabras reservadas (conts.) Después de declarar la constante ya puedes usarla en el [cuerpo principal del programa](http://teleformacion.edu.aytolacoruna.es/PASCAL/document/estruct.htm#cuerpo). Tienen varios usos: ser miembro en una expresión, en una comparación, asignar su valor a una variable, etc.

H. EJEMPLO CONSTANTES Y VARIABLES.

Las constantes son valores inmutables que se conocen en tiempo de compilación y no cambian mientras dura el programa. Las constantes se declaran con el codificador [***const***](https://msdn.microsoft.com/es-es/library/e6w8fe1b.aspx).Solo los tipos integrados de ***C# (excluido [System.Object](https://msdn.microsoft.com/es-es/library/system.object.aspx))*** se pueden declarar como **const**. Para obtener una lista de los tipos integrados, vea [***Tabla de tipos integrados (Referencia de C#)***](https://msdn.microsoft.com/es-es/library/ya5y69ds.aspx)***.***Los tipos definidos por el usuario, incluidas las clases, los ***structs*** y las matrices, no pueden ser ***const.*** Use el modificador ***[readonly](https://msdn.microsoft.com/es-es/library/acdd6hb7.aspx)*** para crear una clase, un ***struct*** o una matriz que se inicialice una vez en tiempo de ejecución (por ejemplo, en un constructor) y que posteriormente no se pueda cambiar.

## I.Pseudocódigo para Instrucciones de Lectura o Entrada de Datos

Las instrucciones de lectura, tienen palabras asociadas de entrada tales como:

* input
* get
* read

En nuestro pseudolenguaje utilizaremos la palabra **leer**, y tiene como argumento, el nombre de una variable con el siguiente formato:

Leer (NombreDeVariable);

Las **Entradas** son un conjunto de valores que pueden estar en un archivo de disco, cinta magnética o proceder del teclado. Para nuestro caso, estas entradas serán dadas. Asociadas a las **Entradas** hay un apuntador que inicialmente apunta al primer valor y cada vez que se lea un valor, este apuntador se mueve un valor para adelante

**I.1 Ejemplo:**

 si las entradas son:

**5 7 10 20**

Y se ejecutan las instrucciones:

Leer(A);
leer (B);

Con la primera instrucción, el procesador hace los siguientes pasos:

1. leer el valor actual de las **Entradas**: 5
2. mover el apuntador de las **Entradas** un lugar adelante: **5 7 10 20**
3. Almacenar el valor leído 5 en la localidad de memoria **A**

Igualmente para la segunda instrucción, se almacenará **7** en **B** y el apuntador estará en **10.**

2. OPERADORES ARITMETICOS

Tal como era de esperarse los operadores aritméticos, mostrados en la **TABLA 4 ,** comprenden las cuatro operaciones básicas , suma , resta , multiplicación y división , con un agregado , el operador módulo .

**TABLA 4 OPERADORES ARITMETICOS**

|  |  |  |  |
| --- | --- | --- | --- |
| **SIMBOLO** | **DESCRIPCION** | **EJEMPLO** | **ORDEN DE EVALUACION** |
| **+** | **SUMA** | **a + b** | **3** |
| **-** | **RESTA** | **a - b** | **3** |
| **\*** | **MULTIPLICACION** | **a \* b** | **2** |
| **/** | **DIVISION** | **a / b** | **2** |
| **%** | **MODULO** | **a % b** | **2** |
| **-** | **SIGNO** | **-a** | **2** |

OPERADORES RELACIONALES

Todas las operaciones relacionales dan sólo dos posibles resultados: **VERDADERO o FALSO**. En el lenguaje C, Falso queda representado por un valor entero nulo (cero) y Verdadero por cualquier número distinto de cero En la TABLA 5 se encuentra la descripción de los mismos.

**OPERADORES RELACIONALES**

|  |  |  |  |
| --- | --- | --- | --- |
| **SIMBOLO** | **DESCRIPCION** | **EJEMPLO** | **ORDEN DE EVALUACION** |
| **<** | **menor que** | **(a < b)** | **5** |
| **>** | **mayor que** | **(a >b)** | **5** |
| **< =** | **menor o igual que** | **(a < = b)** | **5** |
| **>=** | **mayor o igual que** | **( a >>= b )** | **5** |
| **= =** | **igual que** | **( a = = b)** | **6** |
| **! =** | **distinto que** | **( a != b)** | **6** |

OPERADORES LOGICOS
Hay tres operadores que realizan las conectividades lógicas **Y (AND), O (OR) y NEGACION (NOT).**

**OPERADORES LOGICOS**

|  |  |  |  |
| --- | --- | --- | --- |
| **SIMBOLO** | **DESCRIPCION** | **EJEMPLO** | **ORDEN DE EVALUACION** |
| **&&** | **Y (AND)** | **(a>b) && (c < d)** | **10** |
| **||** | **O (OR)** | **(a>b) || (c < d)** | **11** |
| **!** | **NEGACION (NOT)** | **!(a>b)** | **1** |

OPERADORES DE ASIGNACION

En principio puede resultar algo ***fútil*** gastar papel en describir al operador ***IGUAL A ( = ) ,*** sin embargo es necesario remarcar ciertas características del mismo .
Anteriormente definimos a una asignación como la copia del resultado de una expresión ***(rvalue)*** sobre otra ***(lvalue)*** , esto implica que dicho ***(lvalue)*** debe tener **LUGAR** (es decir poseer una posición de memoria ) para alojar dicho valor .
Es por lo tanto válido escribir

Por ejemplo:

a += b ; /\* equivale : a = a + b \*/

a -= b ; /\* equivale : a = a - b \*/

a \*= b ; /\* equivale : a = a \* b \*/

a /= b ; /\* equivale : a = a / b \*/

a %= b ; /\* equivale : a = a % b \*/