**Bases de programación**

**Programación**

La programación se basa en organizar una secuencia de pasos ordenados para seguir cierto orden dado.

Dicho de otra forma, *es el acto de decirle a un dispositivo electrónico que es lo que queremos que haga por nosotros para ahorrarnos ese trabajo*, un ejemplo simple de esto que podríamos considerar es por ejemplo la calculadora en nuestro celular o computadora. Esta ya viene con instrucciones (programadas) que se ejecutaran por medio de nuestras interacciones o cuando sea necesario, podríamos presionar la tecla “uno” y esta se mostraría en la pantalla de la calculadora y si presionamos la tecla de “+” la computadora sabe que lo que tendrá que hacer es una suma, pues estas instrucciones ya se le fueron “dadas” a el dispositivo, de esta forma es que sabe lo que puede hacer.

Si bien nosotros mencionamos una calculadora en una computadora o celular, esto mismo aplica a las calculadoras físicas, pues solo estamos escribiendo instrucciones para que un dispositivo (el hardware y software) pueda leerlas.

**¿Cómo es que logran darle instrucciones a una máquina para que esta la siga?**

Primeramente, debemos de saber que las instrucciones que les damos a la maquinas deben de ser muy precisas, pues es como si le enseñaras a un niño como hacer algo, pues las maquinas seguirán las instrucciones al pie de la letra y si algo no está bien definido entonces tus instrucciones que le diste fallarán.

Para poder evitar esto los programadores lo que hacen es seguir una metodología la que le llamamos “La metodología de solución de problemas”. Es a través de esta que podemos realizar nuestro algoritmo y diagrama de flujo.

Para empezar que pasos son los que necesitamos seguir para la metodología son los siguientes.

**1.-Definimos el problema**

Que es lo que exactamente queremos resolver con nuestro algoritmo, si por ejemplo alguien no esta solicitando que resolvamos el área de un rectángulo, debemos de saber cuáles son las fórmulas necesarias para obtener el resultado o si nos vamos a un ejemplo más complejo, si un cliente(por ejemplo una farmacia) nos pide que hagamos un programa(un software para resolver un problema) en el cual incluyamos los productos que vende, nosotros necesitamos saber cuáles son las características de estos productos y de que forma quiere que los mostremos o en que parte requiere que guardemos toda esta información.

Todos estos datos que le solicitamos al cliente para poder hacer nuestro programa lo más apegado a la visión del cliente es lo que conocemos como una definición de nuestros problemas.

**2.-Analisis del problema.**

Una vez que hayamos definido cuales son los problemas que nuestro cliente quiere que resolvamos pasaremos al paso de análisis, en este paso lo que hacemos es determinar cuáles son las especificaciones de entrada proceso y salida.

Tomando como ejemplo “Como calcular el área de un rectángulo”.

Los **datos de entrada** corresponden a los datos que el cliente va a ingresar para encontrar una respuesta, lo que correspondería a ingresar tanto la altura como ancho del rectángulo

El **proceso** corresponde a los procedimientos que vamos a realizar, ya sean cálculos u organización de la información, Corresponde a saber cuál es la fórmula del área del rectángulo y poder describir como es que se hace, pues la máquina desconoce cómo es que se hace.

Y, por último, la **información de salida** corresponde a la información ya procesada y que por lo tanto la consideraremos el resultado de los datos que hayamos ingresado, en otras palabras, después de haber utilizado la formula para obtener el área del rectángulo, mostraremos el resultado de esta.

**3.-Diseño de la solución.**

En este paso es donde empezaremos a armar nuestra solución al problema, ¿Y cómo lo haremos?, pues es donde aplicaremos nuestro algoritmo y el diagrama de flujo, aunque antes de poder mostrarlo explicaremos que son los tipos de variables y los ciclos.

A partir de este paso los siguientes corresponden a lo que seria programar en la maquina y otras cosas más allá de temas simples de pseudocódigo, pero pues aun así lo mencionare ya que son parte de esta metodología.

**4.-Codificacion.**

Es donde traduciremos nuestro diagrama de flujo a código.

**5.-Compilacion y ejecución.**

Aquí es donde comprobaremos que nuestro programa este bien escrito y observar si este funciona.

**6.-Verificacion y pruebas.**

Verificamos cualquier tipo de errores que hayamos identificado, ya sea nuestro o de que la lógica en general sea errónea.

**7.-Depuracion.**

Corregimos los errores identificados.

**8.-Documentacion.**

Explicamos en un documento el funcionamiento interno de nuestra aplicación (variables utilizadas, que proceso se usa, que lenguaje de programación ocupa, etc.) y el funcionamiento externo (que valores ingresamos y que valor nos regresa).

Antes de adentrarnos en los algoritmos y diagramas de flujo, primero tenemos que conocer cuales son los tipos de variables que utilizaremos.

**Variables**

Básicamente tienen la misma definición que en matemáticas, son valores que varían dependiendo de que valor de demos nosotros (en nuestro caso el usuario).

Existen una inmensa cantidad de tipos de datos para las variables que podemos tomar en cuenta y que además estas mismas tienen otras mas variaciones que se pueden utilizar de la misma, pero intentare resumirla en las más genéricas que todos los programadores siempre utilizaran.

**Int**

Este tipo de variable es numérico, es decir que solo se podrá utilizar con números y no se podrán colocar letras o caracteres especiales, además de que solo estos podrán ser números enteros, se pueden realizar todas las operaciones básicas con este tipo de dato, la única cosa que podríamos destacar es que las divisiones que hagamos con este tipo de datos igualmente solo devolverán número enteros, es decir que si dividimos ½ en números “int” en vez de obtener 0.5 ,nos devolvería solo el entero siendo este 0.

Las variaciones de este tipo de dato son para números en decimales, siendo algún ejemplo de estos datos en float (solo muestra algunos decimales) o double (muestra la mayoría de los números decimales) y por último Decimal (máxima precisión).

**String**

Este tipo de dato nos permite guardar cadenas de texto, es decir, que los datos que guardaremos serán considerados como solo texto, por ejemplo, si escribimos números que estén guardados en una variable string, estos serán considerados solo como texto y desprovistos de su valor numérico.

La variación de este tipo de dato para texto son los datos Char.

**Char**

Es una variación de string que igualmente se encuentra enfocado a texto, pero de una manera mas simple pues este tipo de dato se encuentra enfocado solo a contener o guardar solamente una letra o un solo símbolo.

**Date**

Este tipo de dato es exclusivamente para guardar fechas, este se guardará de la siguiente forma “DD/MM/YY” que corresponde al día, mes y año (Están escritos en inglés, por eso el año se representa con YY que es year).

Existen otras variaciones de este dato que podríamos mencionar como el “datetime” que es para fechas y tiempo, también podríamos solo ocupar el time o incluso existe el TimeStamp que es incluso aun mas preciso pues incluye las fracciones de segundo.

**Bool**

Este dato a diferencia de los antes mencionados es algo especial pues lo utilizaremos para determinar el valor de una decisión, estos valores solamente pueden ser “true” o “false”, por ejemplo, si nosotros vamos a tomar una decisión, el programa lo tomara a modo de pregunta y dependiendo de la respuesta generara un valor ya sea “true” o “false” y esto es lo que nos devolverá el dato.

Se utiliza principalmente en los ifs que es algo que explicaremos más adelante.

**ALGORITMOS**

No son mas que la serie de pasos a seguir para poder resolver un problema.

Los algoritmos los utilizamos para poder crear la estructura de lo que sería de la forma en que creemos que podemos solucionar el problema, aquí lo que haremos será determinar la serie de pasos a seguir para poder llegar a nuestra solución de la mejor manera posible, es decir, que no solo vasta con que solucionemos el problema, sino que también esta solución debe de ser rápida y eficaz.

La razón por la que deberíamos de utilizar los algoritmos es para poder tener una idea, un camino o en general algo a seguir para no tener que saltar directamente a programar en código, estos algoritmos nos permiten armar de forma simple y poder detectar cualquier error o lógica que podría llegar a tener nuestro producto final.

La ventaja de hacer nuestro algoritmo es que cualquier persona puede entender como funcionara nuestro programa final y que además como este es fácil de leer también será más fácil identificar si es que existe algún error.

Ahora aprenderemos como crear nuestro algoritmo.

Como ejemplo tomaremos el de calcular el área de un rectángulo.

1.- inicio

2.- leer el valor de la base del rectángulo.

3.-Leer el valor de la altura del rectángulo.

4.-Calcular el área multiplicando base por altura

5.-Mostrar el valor del área

6.-Fin

Como puedes observar para poder iniciar nuestro algoritmo escribimos la palabra “inicio”, todos los algoritmos siempre iniciaran de esta forma, más que nada para que sea comprensible donde es que se empieza a leer el algoritmo.

En el paso 2 y 3 leemos u obtenemos los valores que son la base y altura pues los ocuparemos para realizar la operación siguiente.

Nótese que ocupamos una palabra clave que es leer para poder denotar que estamos solicitando datos, si bien puedes seguir esta “sintaxis” también podrías utilizar cualquier sinónimo u otras palabras que nos quieran indicar que queremos obtener un dato del usuario. Esto es algo que sucederá también con otras frases como para calcular o para mostrar datos.

En el paso 5 igualmente pasa lo que ya había mencionado antes, no es necesario utilizar las mismas palabras para mostrar datos, pero mientras utilices algo que signifique lo mismo está bien.

Y ya por último en el paso 6 cerramos nuestro algoritmo.

Este ejemplo fue algo corto debido a lo sencillo del problema a resolver, pero si en dado caso nuestro algoritmo llegara a ser mas avanzado podríamos encontrarnos con ciertas situaciones especiales donde se requiere tomar una decisión o de repetir ciertos pasos para llegar al resultado, al caso especial de decisiones ocuparemos la sentencia “si (colocamos nuestra condición) entonces” si esta condición se cumple lo que pasara es que ejecutaremos la instrucción dentro de esta y si no se cumple la condición se coloca la siguiente sentencia “ en caso contrario” y seguimos con el código.

Y el otro caso especial son para los ciclos que, dicho de otro modo, son repeticiones de una instrucción hasta que se cumpla una condición.

Tome por la palabra condición a una decisión que se tomara y que dependiendo de la respuesta se realiza una diferente parte de nuestro algoritmo.

Ahora mostremos un ejemplo de la sentencia si-entonces.

1.-Inicio

2.-Leer base

3.-Leer altura

4.\_**Si** la base > 0 Y la altura > 0 **Entonces**

4.-Calcular área como base × altura

4.1-Mostrar "El área del rectángulo es: " + área

5.**En caso contrario**

5.1-Mostrar "La base y la altura deben ser valores positivos."

5.2-Fin Si

6.-Fin

**Ahora entraremos en temas de ciclos.**

**FOR**

Como ya había mencionado antes los ciclos nos ayudan a repetir cierto conjunto de instrucciones las veces que nosotros especifiquemos o que sean necesarias, por ejemplo, si necesitáramos mostrar la tabla de multiplicar del 2 hasta la décima multiplicación en vez de nosotros escribir por separado cada una de las multiplicaciones podemos utilizar un ciclo **for** para ir haciendo una suma de 2 por el numero anterior y mostrar el resultado.

Se entenderá de mejor forma en el siguiente ejemplo.

1.-Inicio

2.-**para** **i=1 hasta 10 hacer**

2.1- resultado = 2 \* i

2.2 mostrar resultado

2.3.- **fin**

3.-fin

La forma en que funciona nuestro for es la siguiente:

La variable de control “i” es la que ira aumentando con cada iteración hasta llegar al tope que nosotros hemos determinado.

El valor que nosotros le demos es a partir de que numero iniciará la cuenta y el numero que determinará cuando se termina será el que se escribe después de “**hasta**” y la instrucción “**hacer**” es para marcar el inicio del ciclo.

Algo a remarcar es que el incremento se hace automáticamente de 1 en 1 pero podemos modificar esto agregando nosotros un paso donde se determine el incremento

**WHILE**

Siendo su traducción “**mientras**”, su funcionamiento es bastante parecido al del **for**, pero con la diferencia de que en este no tendremos la opción de decir cuantas veces tendremos que repetir las instrucciones si no que se repetirán hasta que se cumpla la condición, por ejemplo, si nuestro usuario ingresa un valor que nosotros no consideramos correcto, este ciclo permitirá que se repita constantemente hasta que este escriba como requerimos el dato. Cabe decir que el ejemplo anterior es algo avanzado para principiantes por lo que utilizaremos el mismo ejemplo en el **for**.

1.-Inicio

3.- i = 1

3.-**mientras (i <= 10) hacer**

3.1-resultado = 2 \* i

3.2 mostrar resultado

3.3 i = i +1

3.4 **Fin**

4.-Fin

El cambio mas notable de este ciclo es que ahora nosotros definimos el incremento (antes era opcional) ademas de que agregamos nuestro incremento en el paso 3.3.

**Do while.**

Siendo su traducción “**has - mientras**”, este ciclo seguirá la misma estructura que los anteriores, pero con la diferencia de que primero se ejecutará la instrucción y luego se considerará la condición.

Básicamente podemos darle el mismo ejemplo de funcionamiento que el **while**, pero con una diferencia que es que mientras el **while** ejecutara la comprobación solo si no se esta cumpliendo la condición nuestro **do-while** la ejecutara al menos una vez.

Ejemplo

1.-Inicio

2.-i=1

3.- **hacer**

3.1-resultado = 2\*i

3.2 mostrar resultado

3.3- i = i +1

3.4 **mientras (i <= 10)**

4.-fin

Con esto hemos finalizado un aprendizaje básico de algoritmos así que ya deberías de poder hacerlo por tu cuenta.

Ahora pasaremos a algo un poco más complejo…

**DIAGRAMAS DE FLUJO**

*¿Para empezar que es un diagrama de flujo?*

Es una representación gráfica de nuestro algoritmo, es decir, que por medio de figuras (también conocidas como símbolos) ya definidas mostraremos el funcionamiento de nuestro algoritmo.

*¿pero porque convertimos el algoritmo a diagrama de flujo?*

Respuestas puede haber muchas, pero a la que yo más me apego es que lo hacemos porque de esa forma podemos visualizar la información lo que equivale a poder comprenderla de mejor forma y poder identificar si esta es correcta y transformarla a código.

Podemos hacer básicamente todo lo mencionado en nuestro tema de algoritmos, pero ahora utilizando figuras como se muestra a continuación.

**­Ovalo:**

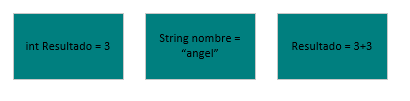
Lo utilizamos para marcar nuestro inicio y fin del diagrama.



**Rectángulo:**

Sera lo que marque nuestros procesos, es decir cuando creamos una variable o realizamos alguna operación.

Una cosa que debemos remarcar es que ahora tendremos que marca de que tipo serán nuestras **variables**.



**Romboide:**

Este lo utilizaremos para marcar el ingreso de datos que nosotros le solicitemos al usuario, es el equivalente de la sentencia “**leer**”

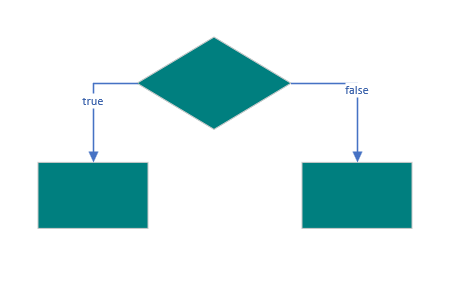


**Cuadrado con una base en forma de S:**

Este lo utilizamos para imprimir cualquier tipo de datos, es el equivalente de la sentencia “**mostrar**”

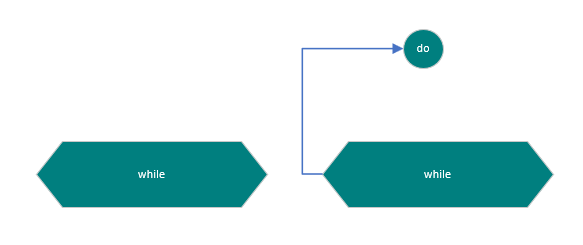


**Rombo:**

Este lo utilizaremos para crear nuestras decisiones (**if**) y equivale a usar la sentencia **si-entonces**

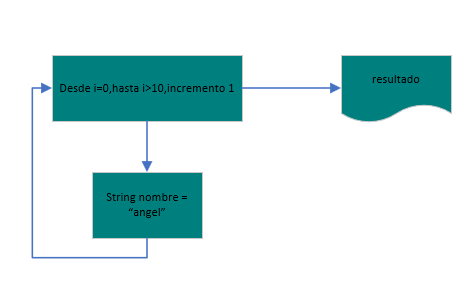
Tendrá dos salidas dependiendo si el resultado es falso o verdadero

**Rectángulo con los lados estirados:**

Este lo utilizaremos para definir el **while** y do **while**.

**Rectángulo (FOR):**

El **for** lo definiremos a partir de un rectángulo de proceso, pero con la siguiente forma de escribirlo:

**---- Desde i = 0, hasta i > 10, incremento 1 ----**

Por último, algo exclusivo de los diagramas de flujo son las “Flechas de flujo” que serán lo que nos va guiando en donde seguir nuestro diagrama como ya habras visto en algunos de los ejemplos anteriores.

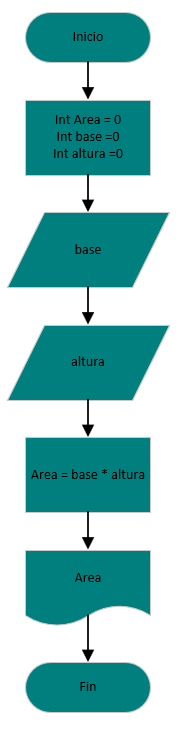
Ejemplos de diagrama de flujo:

Considero que con la información aprendida del tema de algoritmos es suficiente para poder crear diagramas de flujo pues su funcionamiento es el mismo, solamente habría una cosa que aclarar y eso es la “**declaración de variables”.**

Hacemos la declaración de variables al principio de nuestro diagrama justo después de nuestro inicio.

*¿Por qué hacemos esto?*

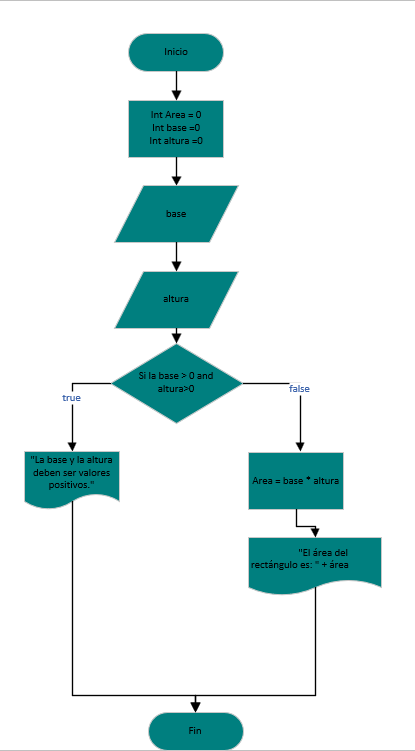
Lo hacemos porque así tenemos en claro que variables utilizaremos y cuáles serán sus tipos de datos, nos ayuda tener una mejor visión de que haremos con nuestro diagrama además de que así se ve mas estético.

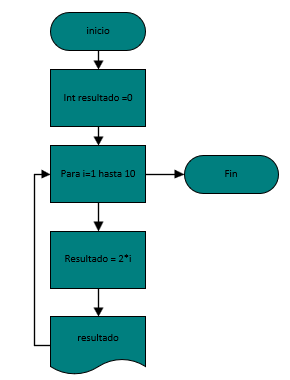
Véase el siguiente ejemplo.

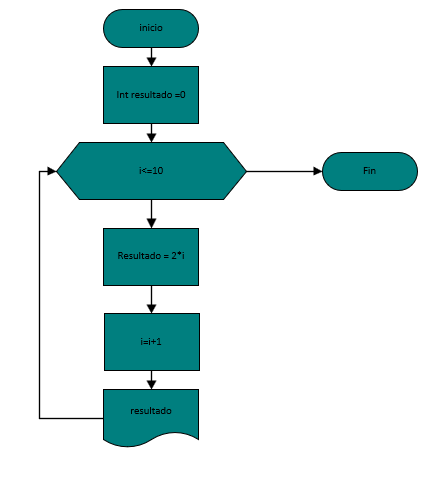
Aquí podemos ver nuestra primera diferencia con nuestro algoritmo que es de *declarar nuestras variables.*

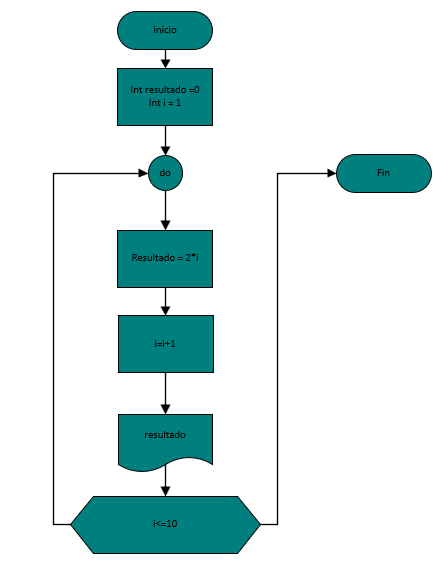
Por lo demás es básicamente lo mismo, solo resta seguir el diagrama hasta el final.

Algo que debería de mencionar es que cuando compruebas si tu diagrama funciona realizando lo que se dice en este a mano, se le conoce como “**prueba de escritorio”**

**Diagrama para el si-entonces(if).**

**Diagrama para el ciclo for.**

**Diagrama para el ciclo while.**

**Diagrama para el ciclo do-while.**

¡Y listo con esto podemos dar por terminado nuestro aprendizaje básico para poder adentrarte al mundo de la programación!

Algunos temas que deje de lado por si quieres checar por tu cuenta.

* Correcta escritura de variables
* Como escribir de modo escrito operaciones
* No incluí todas las posibles formas del diagrama de flujo, existen más por si quieres investigarlo.
* Poo y Poe (son formas de programar, es algo avanzado).