|  |  |  |
| --- | --- | --- |
|  | **Carátula para entrega de prácticas** | |
| Facultad de Ingeniería | | Laboratorio de docencia |

Laboratorios de computación

salas A y B

|  |  |
| --- | --- |
| *Profesor:* | Rodriguez Espino Claudia |
| *Asignatura:* | Fundamentos de Programacion |
| *Grupo:* | 1102 |
| *No de Práctica(s):* | Practica numero 4 |
| *Integrante(s):* | Gutierrez Orozpe Luis Fernando |
|  |  |
|  |  |
| *Semestre:* | 1er Semestre. |
| *Fecha de entrega:* | 10 de Septiembre de 2017 |
| *Obervaciones:* |  |
|  |  |

CALIFICACIÓN: \_\_\_\_\_\_\_\_\_\_

OBJETIVO

Elaborar diagramas de flujo que representen soluciones algorítmicas vistas como una serie de acciones que comprendan un proceso.

ACTIVIDADES

Durante la práctica pudimos analizar las diferentes partes de las que se compone un diagrama de flujo y las aplicaciones que le podemos dar a cada una. Los diagramas de flujo representan de manera gráfica un proceso, es decir, muestran gráficamente el flujo de acciones a seguir para cumplir con una tarea específica.

Lo anterior se traduce, dentro de las ciencias de la computación, que un diagrama de flujo es la representación gráfica de un algoritmo. Saber construir bien un diagrama de flujo es esencial para la etapa de la codificación en la programación, ya que, a partir del diagrama es posible codificar un programa en algún lenguaje de programación.

Un diagrama de flujo tiene diferentes formas, por ejemplo, los óvalos representan el inicio y el fin del diagrama de flujo, las líneas son utilizadas para indicar la dirección del diagrama y estas deben ser rectas, verticales u horizontales. A todo símbolo en el diagrama de flujo, le corresponde solo una línea de dirección de flujo, si dos líneas de dirección están tocando a un símbolo, se considera una operación invalida, las líneas solo se pueden unir entre sí.

Una vez que analizamos y vimos un poco de teoría acerca de cómo son los diagramas de flujo y cuáles son sus diferentes formas, entramos al análisis de las Estructuras de control de flujo, es en esta parte en la que nos vamos a centrar un poco más ya que, dentro de esta, podremos entender las actividades que se realizaran más adelante.

**Estructuras de control de flujo.**

Las estructuras de control de flujo permiten la ejecución condicional y la repetición de un conjunto de instrucciones. Existen 3 estructuras de control las cuales son: secuencial, condicional y repetitivas o iterativas.

**Estructura de control secuencial**

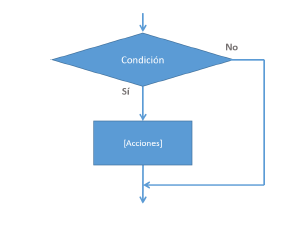
Las estructuras de control secuenciales son las sentencias o declaraciones que se realizan una a continuación de la otra en el orden en el que están escritas.



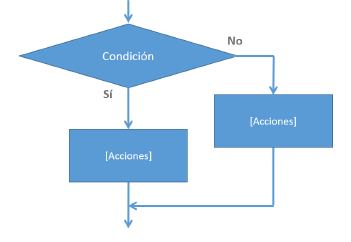
**Estructura de control condicional (o selectiva)**

Las estructuras de control condicionales permiten evaluar una expresión lógica (condición que puede ser verdadera o falsa) y, dependiendo del resultado, se realiza uno u otro flujo de instrucciones. Estas estructuras son mutuamente excluyentes (o se ejecuta una acción o se ejecuta la otra)

La estructura de control de flujo más simple es la estructura condicional si (IF), su sintaxis es la siguiente:



Se evalua la expresión lógica y si se cumple (si la condición es verdadera) se ejecutan las instrucciones del bloque (acciones). Si no se cumple la condición, se continua con el flujo normal del programa. La estructura completa de este tipo es SI-DE LO CONTRARIO (IF-ELSE):

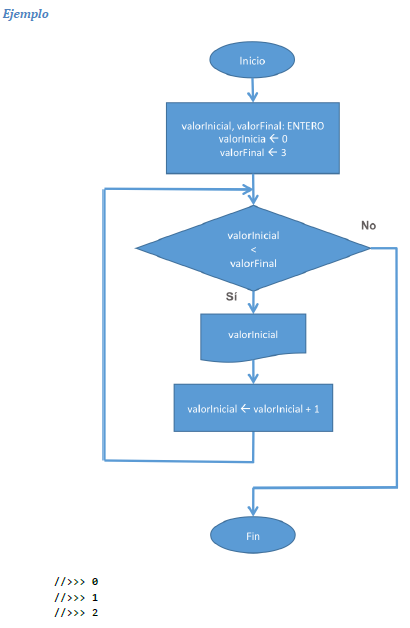


Es con este tipo de estructuras con el que trabajaremos para la práctica ya que, los problemas que tenemos que resolver dependen de condiciones para que puedan llevarse a cabo, de lo contrario, sería imposible poder encontrar una solución óptima.

**Estructuras de control iterativas o repetitivas.**

Las estructuras de control de flujo iterativas o repetitivas, permiten ejecutar una serie de instrucciones mientras se cumpla la expresión lógica. Existen dos tipos de expresiones cíclicas MIENTRAS Y HACER-MIENTRAS.

La estructura MIENTRAS primero valida la condición y si esta es verdadera procede a ejecutar el bloque de instrucciones de la estructura, de lo contrario rompe el ciclo y continua el flujo normal del programa



En el ejemplo anterior se puede observar cómo se va a desarrollar un ciclo, siempre y cuando se cumpla la condición de que el valor inicial sea menor que el valor final. Por esta razón, en la parte inferior del ejemplo, el ciclo termina en el número 2, ya que al ser 3 el número que sigue en el ciclo, la condición se vuelve falsa y entonces se llega al fin del programa.

A continuación se mostraran una serie de problemas a resolver en donde se tendrá que realizar el algoritmo y posteriormente hacer su representación gráfica (diagrama de flujo) para comprobar la validez del algoritmo.

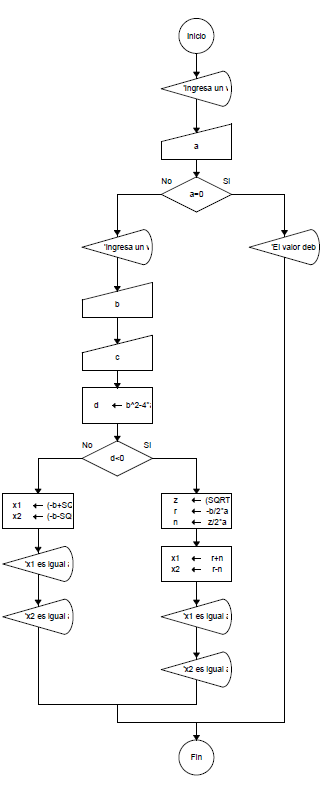
**PROBLEMA 1**

Se quiere resolver la ecuación cuadrática de manera que muestre tanto raíces reales como raíces imaginarias, de la misma forma, se debe pedir al usuario que ingrese un valor de la variable “a” diferente de 0, ya que de lo contrario se causaría una indeterminación y se terminaría el programa.

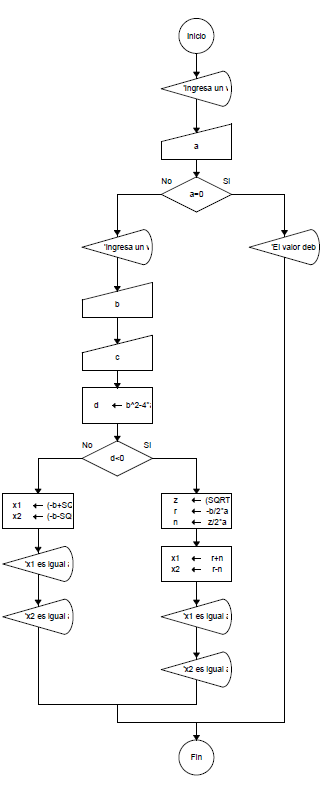
**ALGORITMO**

1. INICIO
2. Pedir el valor para “a” que sea diferente de cero.
3. Si “a”=0 mandar mensaje “No es posible porque crea una indeterminación” y regresar al paso 2; en caso contrario continuar al paso 4.
4. Pedir un valor para “b”
5. Pedir un valor para “c”
6. Realizar operación b2 – 4ac = d
7. Si d < 0, entonces realizar operación = z, hacer la operación **r = - b / 2a**, **n=z/2a** imprimir **x1= r + ni** y **x2= r - ni**, en caso contrario ir al paso 8.
8. X1= x2=
9. Mostrar x1 y x2
10. Fin

**DIAGRAMA DE FLUJO**

****

En esta primera parte del diagrama de flujo, muestra los primeros 5 pasos del algoritmo. Pide al usuario ingresar un valor “a” diferente de cero, de lo contrario termina el programa y no lo deja avanzar a los siguientes pasos que consisten en ingresar las variables “b y c”.



En esta segunda parte, podemos observar cómo es que hay una línea de lado derecho que nos lleva directo al fin del programa, esto es debido a que se le pidió al usuario ingresar un valor de “a” diferente de 0, por lo que, mientras no cumpla esa condición, el programa lo “rechazara”.

Por otro lado, podemos ver que de lado izquierdo están los procedimientos que seguían cuando la condición principal se cumplía. Primero obtenemos un valor de “d” que se obtiene al realizar la ecuación señalada. Posteriormente nace una nueva condición en la que si el valor de “d” es menor que 0, entonces se obtendrán raíces imaginarias, por lo que se realizaran más procesos para obtener dichas soluciones, de lo contrario, se obtendrán raíces reales y solo se realizara un proceso.

En el caso anterior tenemos un ejemplo de Estructuras de control condicionales (al momento de pedir al usuario un valor de “a” diferente de 0) y también estructuras de control secuenciales (cuando vamos a obtener los valores de x1 y x2).

**PROBLEMA 2**

Se le pide al usuario ingresar 3 valores, de estos tres valores el programa debe analizar lo siguiente:

-Si 3 lados son iguales, se formara un triángulo equilátero

-Si 2 lados son iguales y uno diferente, se forma un triángulo isósceles.

-Si los 3 lados son diferentes se formara un triángulo escaleno.

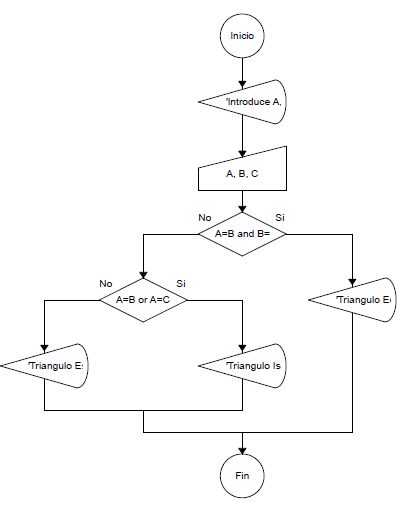
**ALGORITMO**

1. INICIO
2. Pedir valores para A, B y C.
3. Si A=B y B=C, mandar mensaje ‘Triángulo Equilátero’ e ir al paso 5, de lo contrario ir al paso 4.
4. Si A=B o A=C o B=C, mandar mensaje ‘Triangulo Isósceles’ e ir al paso 5, de lo contrario, mandar mensaje ‘Triangulo Escaleno’ e ir al paso 5.
5. FIN

**DIAGRAMA DE FLUJO**

Al realizar el diagrama de flujo podemos comprobar que cuando la condición de que A=B y B=C, nos manda un mensaje de que el triángulo es Equilátero y por lo pronto termina con el programa.

De lo contrario, manda la condición de que si A=B o A=C o B=C entonces nos muestra el mensaje ‘Es un triángulo Isósceles’ y posteriormente termina con el programa, pero si no se cumple con esta condición, entonces será un triángulo Escaleno.

****

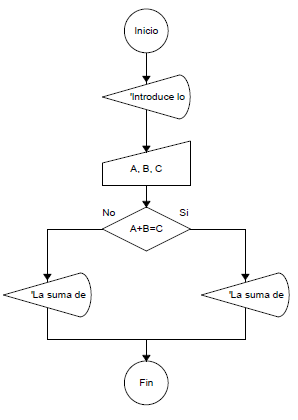
**PROBLEMA 3**

Se le pide al usuario ingresar 3 cantidades y el programa debe analizar si la suma de las dos primeras cantidades ingresadas es igual a la tercera cantidad ingresada

**ALGORITMO**

1. INICIO
2. Ingresa los valores de A, B Y C.
3. Si A+B=C mostrar mensaje ‘La suma de las dos primeras cantidades es igual a la tercera’ e ir al paso 4, de lo contrario, mostrar mensaje ‘La suma de las primeras dos cantidades no es igual a la tercera’ e ir al paso 4.
4. FIN

**DIAGRAMA DE FLUJO**

****

En este ultimo ejemplo, el diagrama de flujo comprueba la condición de que si A+B=C el programa nos enviara el mensaje “La suma de las dos primeras cantidades es igual a la tercera” y finalizara el programa, de lo contrario, nos mostrara el mensaje “La suma de las primeras dos cantidades es diferente de la tercera” y finalizara el programa.

**CONCLUSIONES**

Esta práctica fue mucho de mi agrado ya que estamos reforzando los conocimientos adquiridos al momento de realizar algoritmos para poder solucionar problemas, además de que los diagramas de flujo son una buena herramienta que nos ayuda a darle validez a nuestro algoritmo.

Pienso que el diagrama de flujo, al igual que el algoritmo, son partes fundamentales al momento de solucionar un problema o realizar un programa, por lo que me emociona poder poner el practica estas técnicas para que en el futuro en que veamos programación, pueda realizar las actividades de una manera más fácil y eficiente.