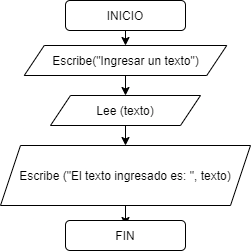
Laboratorio 1. Primeros pasos con Dev C/C++

**Ejercicio 1**

Implementar un programa que pida un texto al usuario y lo escriba por pantalla.

* Diseño del algoritmo.



* Cómo se llega a la solución.

Para llegar a la solución primero se debe recoger los requerimientos del problema:

* Solicitar que el usuario ingres un texto.
* Que lo ingrese por pantalla.
* El usuario pueda leer el texto introducido.
* Fin.
* Tipos de estructuras utilizadas.

Se utiliza la Función printf, que forma parte de la biblioteca estándar de funciones de entrada y salida.

* Justificación del uso de dichas estructuras.

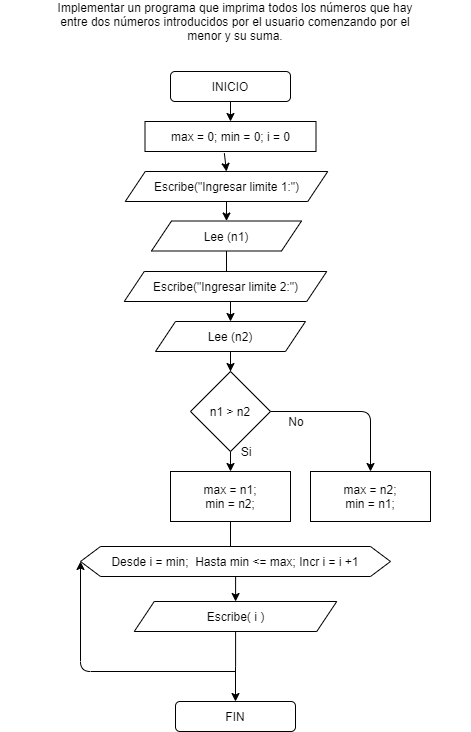
La función printf, nos permite mostrar mensajes de texto en la pantalla cuando ejecutamos un programa.

* Comentarios sobre los posibles errores.

**Ejercicio 2**

Implementar un programa que imprima todos los números que hay entre dos números introducidos por el usuario comenzando por el menor y su suma.

* Diseño del algoritmo.



* Cómo se llega a la solución.

Para llegar a la solución primero se debe recoger los requerimientos del problema:

* Solicitar que el usuario ingrese dos valores enteros
* Que lo ingrese por pantalla.
* Ordenar número mayor y número menor.
* Determinar los numero consecutivos desde el menor al mayor.
* Imprimir en pantalla todos los números entre los enteros ingresados por el usuario.
* Fin.
* Tipos de estructuras utilizadas.

La estructuras utilizadas son: if- else y un for

* Justificación del uso de dichas estructuras.

If-else: Se utilizó un if-else para determinar que numero ingresado por teclado es el mayor y el menor.

For: Se utiliza el for para recorrer desde el menor o mínimo incrementando +1 hasta llegar al valor máximo o el número mayor. Y así poder imprimir los números que se encuentran en ese rango de valores min y max.

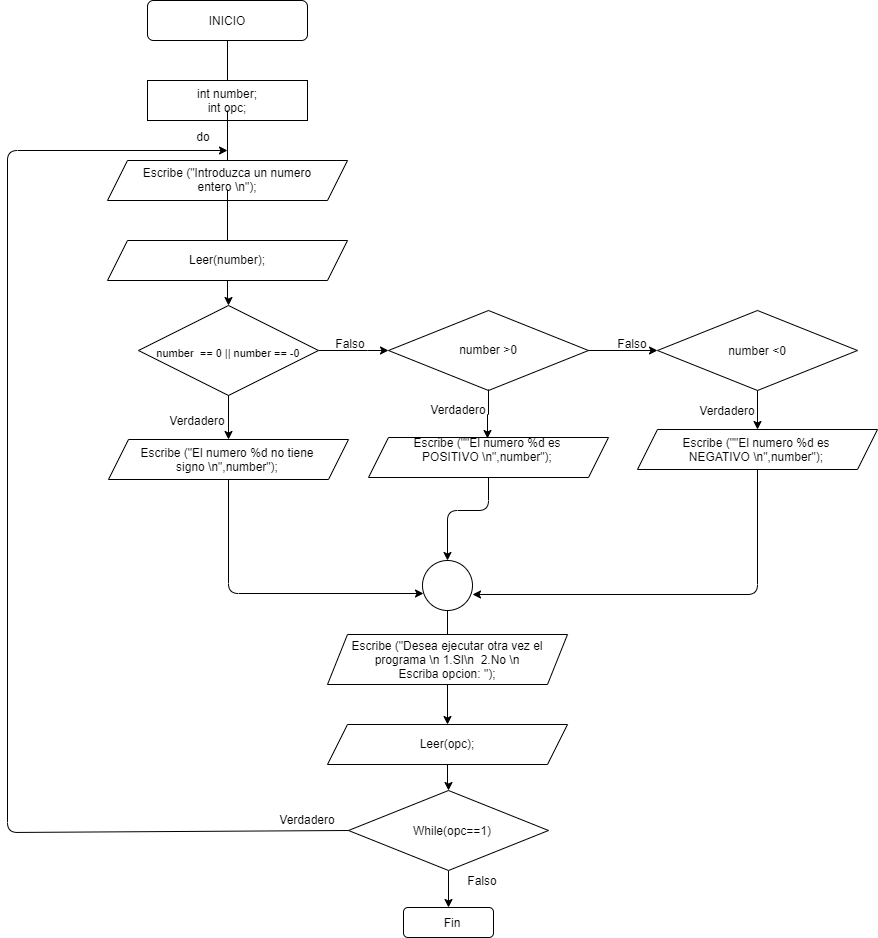
* Comentarios sobre los posibles errores.

-Sin comentario.

**Ejercicio 3**

Implemente un programa que solicite un número y escribir por pantalla si es positivo o negativo.

* Diseño del algoritmo.



* Cómo se llega a la solución.

Para llegar a la solución primero se debe recoger los requerimientos del problema:

* Pedir un número entero.
* Saber si el número es positivo o negativo o cero.
* Imprimir si el numero ingresado por teclado es positivo o negativo utilizamos un if.
* Repetir el bucle hasta que no desee ingresar más números enteros utilizamos el bucle repetitivo do-while.
* Tipos de estructuras utilizadas.

En este problema se utiliza la estructura ***If-else*** y ***do-while***.

* Justificación del uso de dichas estructuras.

***If-else:*** para validar o verificar si el numero entero ingresado es positivo, negativo o cero y dependiendo el resultado imprimir en pantalla el texto “El número… es un numero negativo /positivo/cero”.

***do-while:*** Para que el ciclo if se repita hasta que ya no desee ingresar más números enteros. Se valida de la siguiente manera while(opc==1): si la opc es 1 es decir es verdadero volver hacer el ciclo if, si no es verdadero salir del bucle do-while, y cerrar el programa.

* Comentarios sobre los posibles errores.

Se puede ingresar cualquier número que no sea 1 para terminar el bucle. Y salir del programa, no es necesario que sea un 2.

**Ejercicio 4**

Implementar un algoritmo que cuente el número de caracteres captados desde teclado, determinar si es un digito o un carácter o un espacio, e imprimir por pantalla estos números según el tipo.

* Diseño del algoritmo.

Diagram

Description automatically generated

* Cómo se llega a la solución.

Para llegar a la solución de este problema primero se realiza el análisis de los requerimientos que son:

-Escribir por teclado un espacio, un dígito o una letra.

-Determinar si lo escrito por teclado es un espacio, un dígito o una letra e imprimir en pantalla el resultado.

-Obtener un contador por cada letra, espacio y digito, y mostrar dicho contador en la pantalla.

* Tipos de estructuras utilizadas.

Las estructuras utilizadas son***: if-else*** y un ***do-while***.

* Justificación del uso de dichas estructuras.

***If-else:*** se utiliza esta estructura para validar si el char ingresado por el teclado es primero una letra mayúscula o minúscula si es verdad imprime a la pantalla “La letra… es minúscula/mayúscula” e incrementa en +1 al contador de carácter, si es falso esta instrucción continua con el siguiente if para validar si es un espacio igualmente imprime “Es un espacio” e incrementa en +1 al contador de espacios; Y si esto fuera falso solo está el else para el digito igualmente imprime “El .. es un digito”, e incrementa un +1 al contador de digito.

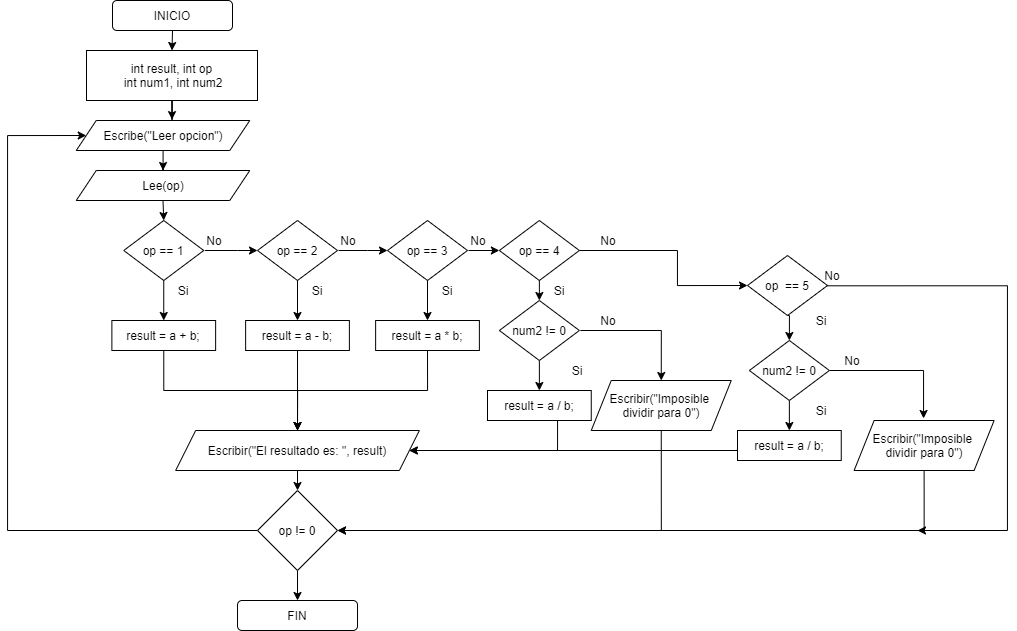
***Do-while:*** Se utilizó para repetir los if anidados es decir volver a realizar la ejecución del programa principal. Se pregunta al usuario si desea volver a escribir otra vez, si la respuesta es 1 el bucle do-while(es verdadero) se repite y si es falso sale del programa y finaliza.

* Comentarios sobre los posibles errores.
* Posibles errores no se validaron si se ingresa un carácter diferente a una letra, espacio o número.
* Para salir del programa se puede escribir cualquier otro número que no sea el 1.

**Ejercicio 5**

Implementar un programa que muestre un menú al usuario para que seleccione la operación que desea realizar: suma, resta, división, módulo o multiplicación y según la opción seleccionada se realice la operación solicitando previamente los números al usuario por teclado.

* Diseño del algoritmo



* Cómo se llega a la solución.

Para resolver este problema es importante conocer las operaciones básicas de C como multiplicación, división, suma, resta y modulo. Leyendo cuatro variables de tipo entero *op* será utilizada para seleccionar una opción del menú, *num1 y num2* serán utilizadas para hacer las operaciones con números y *result* almacenara el resultado de cada una. Para seleccionar la opción usamos una sentencia de control anidada o *switch* y lo encapsulamos dentro de un *do-while* para repetir el programa hasta que la opción sea 0.

* Tipos de estructuras utilizadas.

Las estructuras de control que utilizamos fueron: ***do-while,*** ***switch***, ***if-else.***

* Justificación del uso de dichas estructuras.

***Do-while****:* para repetir el programa las veces que desee el usuario hasta ingresar 0/

***Switch:*** para seleccionar la opción de acuerdo a la operación solicitada por el usuario.

***If-else:*** en el caso del módulo y división es necesario verificar que el segundo número no sea 0.

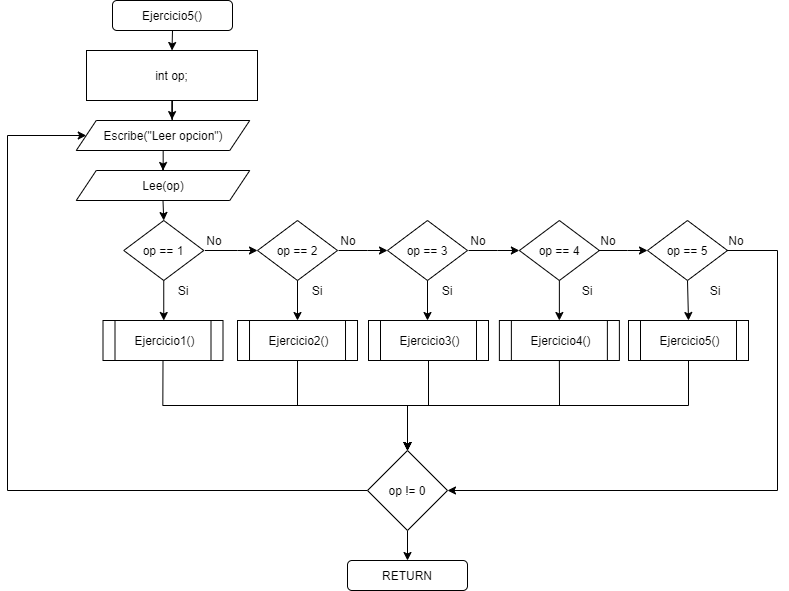
* Comentarios sobre los posibles errores.

Es muy probable que al ingresar caracteres que no sean enteros el programa falle.

**Ejercicio 6**

Finalmente implementar un algoritmo que generar un menú con 5 opciones. La elección de cada opción del menú ejecuta que corresponde a su número dentro del laboratorio.

* Diseño del algoritmo.



* Cómo se llega a la solución.

Para resolver este problema la solución mas apropiada seria importar como funciones cada ejercicio dentro del archivo ***Ejercicio6.c.*** Respectivamente se nombro cada ejercicio como una función de tipo void por ejemplo ***void ejercicio1().***  Que próximamente se usara dentro de una estructura switch con su respectiva opción seleccionada por el usuario.

* Tipos de estructuras utilizadas.

Las estructuras de control que utilizamos fueron: ***do-while,*** ***for, switch***, ***if-else.***

* Justificación del uso de dichas estructuras.

***Do-while****:* para repetir el programa las veces que desee el usuario hasta ingresar 0

***Switch:*** para seleccionar la opción de acuerdo con la operación solicitada por el usuario y ejecutar la función del programa correspondiente.

***If-else:*** Justificado en los programas anteriores.

***For:*** Justificado en el programa 2.

* Comentarios sobre los posibles errores.

Todos los errores que ocasionaban los ejercicios anteriores se verán reflejados en este último.