



**UNIVERSIDAD DE SANTIAGO DE CHILE  
FACULTAD DE INGENIERÍA  
DEPARTAMENTO DE INGENIERÍA INFORMÁTICA**

## **Estructura de Datos y Análisis de Algoritmos**

### **Laboratorio 2**

César Antonio Salazar Silva.

Profesor: Mario Inostroza  
Ayudante: Sebastián Garay  
Franco Leal  
Nicolas Romero  
Daniel Vasquez

Santiago - Chile  
2-2019

## Tabla de Contenidos

CAPÍTULO 1.	Introducción	4
CAPÍTULO 2.	Descripción de la Solución	5
2.1	Marco Teórico	6
2.2	Herramientas y Técnicas	7
2.3	Algoritmos y Estructuras de Datos	7
2.4	Análisis de los Resultados	8
2.5	Conclusión	8
2.6	Referencias	9
CAPÍTULO 3.	Manual de usuario	9
3.1	Introducción	9
3.2	Cómo compilar y ejecutar en linux	9
3.3	Cómo compilar y ejecutar en windows	12
3.4	Funcionalidades del programa	12
3.5	Posibles Errores	12

## Índice de Figuras

Ilustración 1 Descripción formato "imagen.in" .....	5
Ilustración 2 Descripción Formato "Buscar.in" .....	5
Ilustración 3 Encontrar Similitud "Buscar" en "Imagen" .....	5
Ilustración 4 Salida.....	6
Ilustración 5 Matriz ejemplo .....	6
Ilustración 6 Manual 1.....	9
Ilustración 7 Manual revisar archivos. ....	10
Ilustración 8 Manual compilar. ....	10
Ilustración 9 Ejecutar Programa.....	10
Ilustración 10 Ejecutar menú.....	11
Ilustración 11 Introducir nombre Imagen.....	11
Ilustración 12 Introducir nombre imagen a buscar.....	11

## CAPÍTULO 1. INTRODUCCIÓN

Para poder comprender y hacer análisis de algoritmos debemos construir uno y más de uno, dado que con esto podremos estudiar este curso llevando a la práctica los conocimientos teóricos y con ello aprobar la asignatura comprendiendo lo que se ha enseñado efectivamente en ella. Búsqueda de imágenes es la problemática presente en este laboratorio y es por ello que gracias a este tema podemos estudiar lo propuesto para este laboratorio y poder comprender la importancia del Análisis de algoritmos y estructuras de datos ligadas a la solución de un problema y la real “eficiencia” de estos, comparación de algoritmos distintos que a su vez resuelven el problema, pero la gran interrogante es cómo diferenciarlos entre sí, que conceptos hacen una clasificación entre ellos, ¿Será su complejidad, será su tiempo de ejecución, será quizás la cantidad de instrucciones o la cantidad de recursos que utiliza para poder dar solución?, Son un conjunto de interrogantes que se plantean a medida que se le da solución a este laboratorio en paralelo a lecciones entregadas en cátedra que apoyaran la solución o enfoque de este problema. Para este laboratorio se abordará el desarrollo de un algoritmo que permita buscar una imagen dentro de otra a partir de una entrada “imagen.in” y un “buscar.in” y contar cuantas veces está contenida esta imagen y en qué ángulo está presente. Se utilizará el lenguaje de programación C y sus librerías estándar. Además de la experiencia obtenida desde Fundamentos de Programación para dar solución a encontrar una imagen dentro de otra.

## CAPÍTULO 2. DESCRIPCIÓN DE LA SOLUCIÓN

Para dar una solución que dé con la respuesta esperada construiremos un algoritmo que reciba ambos documentos luego de solicitar sus nombres, con ello se piensa hacer una matriz de estructuras "RGB" es decir una estructura de dato con 3 componentes R, G y B con sus respectivos valores leídos a partir del texto y con ello generar 2 matrices, una es en la que se buscará si la segunda está contenida en ella.

```
6 6
0,255,255 255,176,0 0,255,255 0,255,128 255,112,0 6,196,255
0,255,167 255,0,128 0,255,255 255,255,0 255,112,0 7,220,255
0,255,255 255,0,128 0,255,255 255,255,0 214,8,255 7,220,255
6,196,255 7,220,255 7,220,255 7,220,255 255,112,0 7,220,255
255,112,0 255,112,0 214,8,255 255,112,0 255,112,0 7,220,255
0,255,128 255,255,0 255,255,0 7,220,255 0,113,247 253,220,0
```

*Ilustración 1 Descripción formato "imagen.in"*

```
3 4
0,255,128 255,112,0 6,196,255
255,255,0 255,112,0 7,220,255
255,255,0 214,8,255 7,220,255
7,220,255 255,112,0 7,220,255
```

*Ilustración 2 Descripción Formato "Buscar.in"*

Ya con esto realizado se procederá a generar 4 contadores para 0, 90, 180 y 270 los que serán albergados en una lista de cuatro posiciones, la primera para las coincidencias de los pixeles para la rotación de 0, la segunda para la de 90, la tercera para la de los 180 y la cuarta para los pixeles de 270. Creando un arreglo del tamaño de la imagen, se guardará en la posición correspondiente el contador (lista de 4 nodos), a su vez aumentaran en cada ocasión que se encuentre el mismo pixel de buscar,

```
0,255,255 255,176,0 0,255,255 0,255,128 255,112,0 6,196,255
0,255,167 255,0,128 0,255,255 255,255,0 255,112,0 7,220,255
0,255,255 255,0,128 0,255,255 255,255,0 214,8,255 7,220,255
6,196,255 7,220,255 7,220,255 7,220,255 255,112,0 7,220,255
255,112,0 255,112,0 214,8,255 255,112,0 255,112,0 7,220,255
0,255,128 255,255,0 255,255,0 7,220,255 0,113,247 253,220,0
```

*Ilustración 3 Encontrar Similitud "Buscar" en "Imagen".*

Finalmente, para realizar el conteo de mayor, menor similitud se recorrerá esta lista respectivamente a cada Angulo guardando el mayor y el menor elemento. Por otro lado, se van a extraer las cantidades encontradas para encontrar la media, mediana y desviación

estándar de la similitud de los pixeles esto se hará realizando un ciclo que recorra toda la matriz revisando por cada posición la existencia de similitud con la imagen buscada, esto claramente con las restricciones de posición dado que no se puede buscar elementos en posiciones inexistentes. Finalizando con la iteración se espera tener listo los resultados y con ello la salida esperada, la cual será entregada por pantalla, todo esto en un menú que sea un aporte para los ayudantes o profesor (mayor facilidad en la ejecución).

```
Mayor similitud: 6 pixeles encontrado 4 veces
Menor similitud: 2 pixeles encontrado 2 veces
Media: 3.5 pixeles
Mediana: 4 pixeles
Desviacion estandar: 1.976
```

*Ilustración 4 Salida*

## 2.1 MARCO TEÓRICO

Existen distintas formas de afrontar este problema y darle solución, mi propuesta está orientada a crear una matriz con cada imagen así poder iterar con cada posición de la matriz y verificar si existe coincidencia entre sus elementos. Así poder hacerlo para 0, 90, 180 y 270 grados. Esto se hará recorriendo por los índices de diferentes formas y con ello dar la ilusión que se revisa la matriz rotada.

Para esto debemos comprender la forma que se debe iterar en los índices y con ello poder buscar coincidencias para cada rotación.

1 16	5 2	9 3	13 13
2 5	6 11	10 10	14 8
3 9	7 7	11 6	15 12
4 4	8 14	12 15	16 1

*Ilustración 5 Matriz ejemplo*

## 2.2 HERRAMIENTAS Y TÉCNICAS

Para la resolución del problema, es necesario considerar que se trabajará bajo el paradigma Imperativo (procedural), dándole así un conjunto de instrucciones al computador para poder calcular las veces contenidas que hay una imagen en otra. Para el funcionamiento del programa, se utilizan recursos en los algoritmos tal división en sub-problemas (descomposición de un problema en pequeños problemas “más abordables”) y modular en funciones separadas.

## 2.3 ALGORITMOS Y ESTRUCTURAS DE DATOS

En esta sección se explican los dos algoritmos desarrollados, con sus respectivas entradas, salidas y su explicación.

Se hace uso de un structure compuesto de 5 variables int, para: R, G, B, fila y column.

Esta estructura será la albergada en la matriz imagen.

En cuanto al  $t(n)$  podemos ver que el programa es modular y tenemos funciones que analizar.

Para la primera función que se encarga de leer la matriz, siendo esta de orden  $n^2$  y un  $t(n) = 6c + n + 2c + 4cn^2$ .

Para las 4 funciones siguientes (verificar0, verificar90, verificar180, verificar270), el orden en cada una es de  $n^2$  y su  $t(n) = 2c + n^2$ .

Para la función veces contenida, la cual llama a las 4 funciones anteriores debe iterar por largo y ancho a la vez que se utiliza las funciones mencionadas de orden  $n^2$ . De esa forma su orden queda  $n^4$  y su  $t(n) = n^2 * 2 * T(\text{verificar}) + T(\text{promedio}) + T(\text{desviacion}) + T(\text{mediana}) + T(\text{mayorSimilitud}) + T(\text{menorSimilitud})$ . Con un orden final de  $n^4$ .

Cuando se llega al main nos encontramos con el llamado de la función “vecesContenida” y se procede a liberar la memoria utilizada con un orden de  $n^2$  por cada matriz, de esta forma el orden del algoritmo es de  $n^4$ .

Esto, es correspondiente a la primera entrega, para este segundo laboratorio es necesario saber que se utilizaron las siguientes extensiones:

Agregar nodo: Esta función solo agrega el nodo en la cabeza de la lista por lo que su  $T(n) = c$ , por ende, su orden es constante de igual forma.

Ordenar: Esta función recorre la lista, por lo que su orden es “n” y su  $T(n) = 3c + 3cn$ .

Insertar: Dado que recorre la lista hasta agregar el nodo, su orden es de “n” y su  $T(n) = 5c + 1cn$ .

Encontrar Promedio: Como las funciones anteriores solo recorre la lista y su  $T(n) = 3c + 3cn$ .

Encontrar Desviación: Recorre la lista, por lo que su  $T(n) = 4c + 4cn$ , siendo su orden de “n”.

Encontrar Mediana: Algoritmo que recorre la lista 2 veces, pero no de forma anidada lo que deja su Orden de n y su  $T(n) = 4c + 2cn + 2c + 5cn$  en el peor caso.

Similitud Superior: Algoritmo con un  $T(n) = 4c + 2cn + 2c + 3cn$  en el peor caso y un orden de “n”

Similitud Inferior: Algoritmo con un  $T(n) = 4c + 2cn + 2c + 3cn$  en el peor caso y un orden de “n”

## 2.4 ANÁLISIS DE LOS RESULTADOS

Dadas las dos funciones anteriormente mencionadas, se procede a calcular  $T(n)$  y a partir de esto la complejidad del algoritmo

Al analizar estas funciones de acuerdo con los tiempos de ejecución y su complejidad, es claro ver que el algoritmo que más demora es la lectura y el buscar similitudes dentro de las matrices con un orden de “n” a la cuarta.

Sin embargo, también se debe tener en cuenta que el largo de la lista con las similitudes es otro factor a considerar ya que puede ser un arreglo con un largo mayor que el recorrido de la matriz en si y con las funciones de promedio desviación y mediana puede consumir mas recursos en cuanto a tiempo y memoria que un n a la cuarta en leer y buscar similitudes.

## 2.5 CONCLUSIÓN

Ya en este punto solo puedo decir que el algoritmo creado es eficaz pero no puedo decir que sea el más eficiente, tiene un orden de complejidad de  $n^4$  ya que se debe iterar por cada posición buscando coincidencias mientras que itera en la matriz, por lo que el objetivo se cumple,



Se logra cumplir con analizar los algoritmos y las estructuras de datos requeridas al momento que se realiza la solución. Se logra comprender el orden de complejidad de un algoritmo y poder resolver la problemática.

Se debe tener en cuenta que el tiempo de ejecución varía entre el tamaño de la matriz y mientras más grande sea más demora en entregar la solución con un orden de  $n^4$ .

## 2.6 REFERENCIAS

Departamento de Ingeniería Informática, Universidad de Santiago de Chile. (02-04-2019). UdeSantiagoVirtual. (www.udesantiagovirtual.cl) (04-09-2019). [http://www.udesantiagovirtual.cl/moodle2/pluginfile.php?file=%2F258086%2Fmod\\_resource%2Fcontent%2F1%2FEnunciado-Laboratorio-1-2-2019.pdf](http://www.udesantiagovirtual.cl/moodle2/pluginfile.php?file=%2F258086%2Fmod_resource%2Fcontent%2F1%2FEnunciado-Laboratorio-1-2-2019.pdf).

# CAPÍTULO 3. MANUAL DE USUARIO

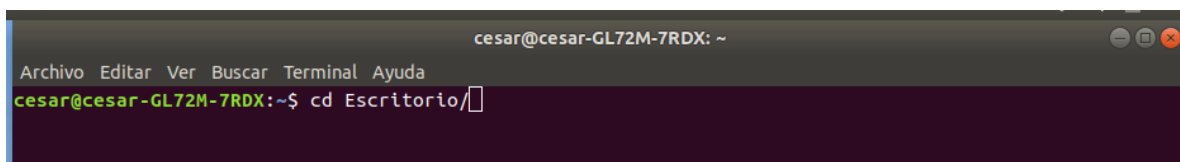
## 3.1 INTRODUCCIÓN

Con el propósito de encontrar una imagen dentro de otra, se ha propuesto este algoritmo de solución que en los siguientes pasos se detalla el modo de uso de manera muy específica.

## 3.2 CÓMO COMPILAR Y EJECUTAR EN LINUX

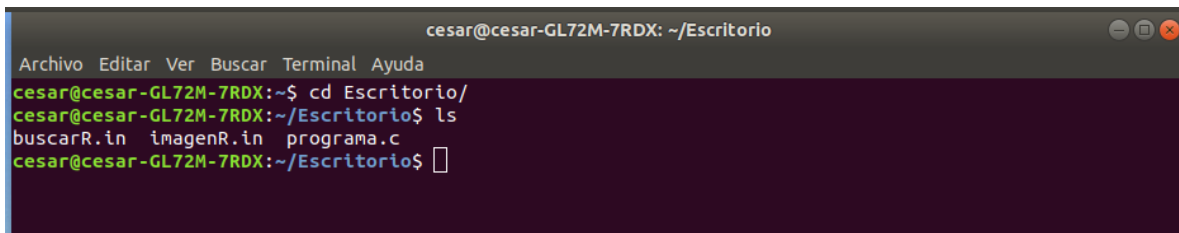
Antes de compilar y ejecutar el programa en el ambiente Linux, es necesario que tanto el archivo que contiene la imagen (“imagen.in”), su imagen a buscar (“buscar.in”), como el archivo que contiene el programa (“programa.c”) se encuentren en el escritorio.

Para compilar el programa, es necesario abrir entonces la terminal, que se puede hacer presionando en el teclado ctrl+shift+T. Una vez aquí, es necesario acceder al escritorio, escribiendo el comando “cd Escritorio” y presionando entonces la tecla Enter como se muestra a continuación:



*Ilustración 6 Manual 1.*

Debería tener lo siguiente: la imagen, la imagen a buscar y el programa.c



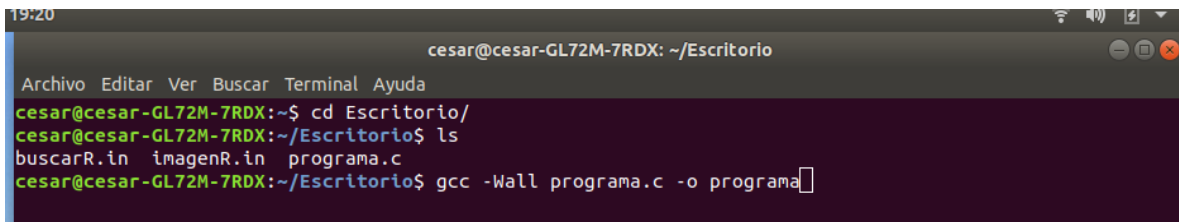
```

cesar@cesar-GL72M-7RDX: ~/Escritorio
Archivo Editar Ver Buscar Terminal Ayuda
cesar@cesar-GL72M-7RDX:~$ cd Escritorio/
cesar@cesar-GL72M-7RDX:~/Escritorio$ ls
buscarR.in  imagenR.in  programa.c
cesar@cesar-GL72M-7RDX:~/Escritorio$

```

*Ilustración 7 Manual revisar archivos.*

A continuación, se debe escribir en la consola el comando “gcc -Wall programa.c -o programa -lm” y volver a presionar la tecla Enter.



```

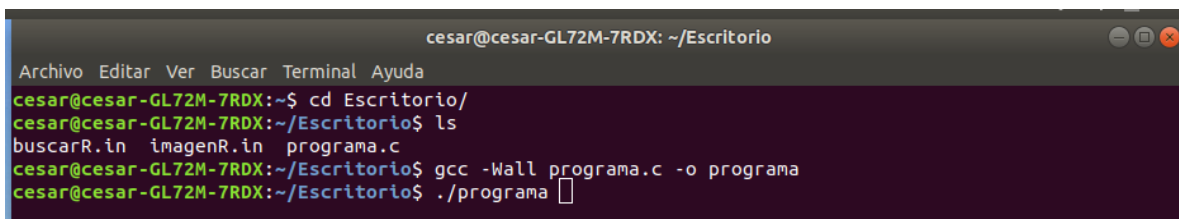
19:20
cesar@cesar-GL72M-7RDX: ~/Escritorio
Archivo Editar Ver Buscar Terminal Ayuda
cesar@cesar-GL72M-7RDX:~$ cd Escritorio/
cesar@cesar-GL72M-7RDX:~/Escritorio$ ls
buscarR.in  imagenR.in  programa.c
cesar@cesar-GL72M-7RDX:~/Escritorio$ gcc -Wall programa.c -o programa

```

*Ilustración 8 Manual compilar.*

Finalizado esto, en el escritorio se crea automáticamente un nuevo archivo dando cuenta de que el programa ya está compilado y a continuación se procede a ejecutarlo.

Para ejecutar el programa, entonces se debe escribir en la consola el comando “./laboratorio” y presionar la tecla Enter. A continuación, se solicitará en consola el nombre del archivo imagen, por lo que deberá ingresar el nombre seguido su extensión como se muestra en la imagen, de la misma forma con el archivo buscar. El programa entregará las coincidencias y en qué ángulos se han encontrado.



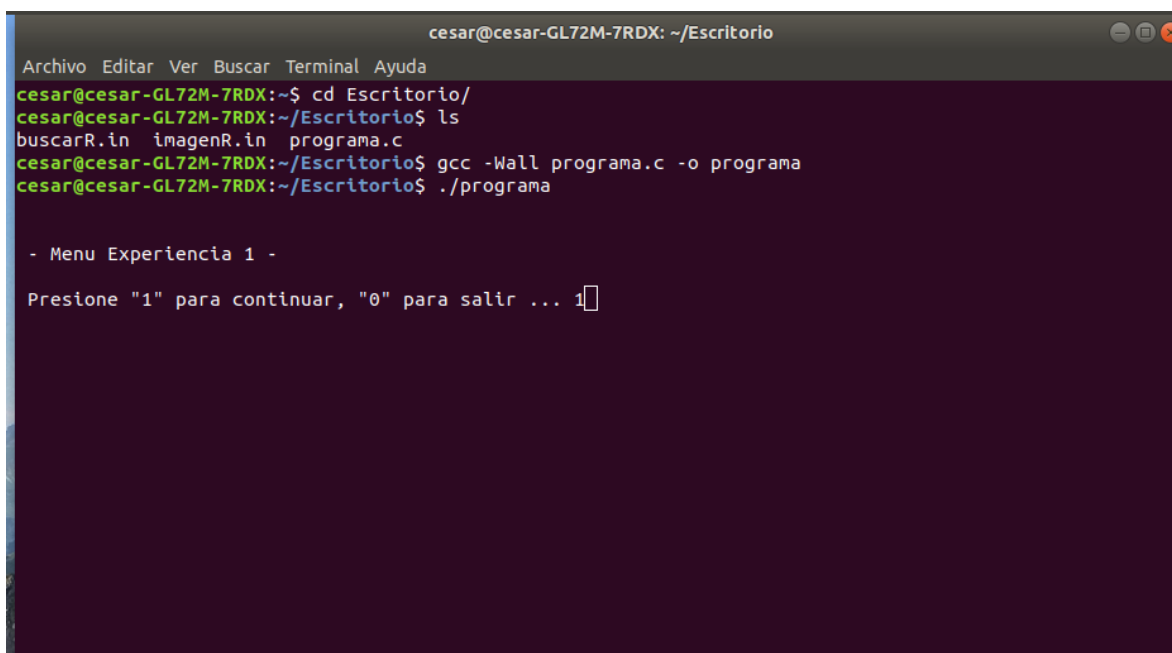
```

cesar@cesar-GL72M-7RDX: ~/Escritorio
Archivo Editar Ver Buscar Terminal Ayuda
cesar@cesar-GL72M-7RDX:~$ cd Escritorio/
cesar@cesar-GL72M-7RDX:~/Escritorio$ ls
buscarR.in  imagenR.in  programa.c
cesar@cesar-GL72M-7RDX:~/Escritorio$ gcc -Wall programa.c -o programa
cesar@cesar-GL72M-7RDX:~/Escritorio$ ./programa

```

*Ilustración 9 Ejecutar Programa*

Se deberá ejecutar de la siguiente forma:



```

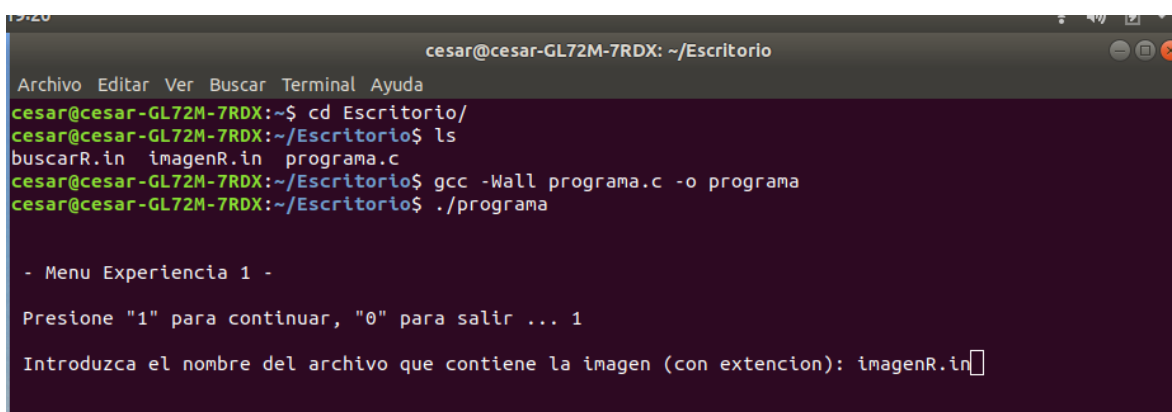
cesar@cesar-GL72M-7RDX: ~/Escritorio
Archivo Editar Ver Buscar Terminal Ayuda
cesar@cesar-GL72M-7RDX:~$ cd Escritorio/
cesar@cesar-GL72M-7RDX:~/Escritorio$ ls
buscarR.in  imagenR.in  programa.c
cesar@cesar-GL72M-7RDX:~/Escritorio$ gcc -Wall programa.c -o programa
cesar@cesar-GL72M-7RDX:~/Escritorio$ ./programa

- Menu Experiencia 1 -

Presione "1" para continuar, "0" para salir ... 1

```

*Ilustración 10 Ejecutar menú*



```

cesar@cesar-GL72M-7RDX: ~/Escritorio
Archivo Editar Ver Buscar Terminal Ayuda
cesar@cesar-GL72M-7RDX:~$ cd Escritorio/
cesar@cesar-GL72M-7RDX:~/Escritorio$ ls
buscarR.in  imagenR.in  programa.c
cesar@cesar-GL72M-7RDX:~/Escritorio$ gcc -Wall programa.c -o programa
cesar@cesar-GL72M-7RDX:~/Escritorio$ ./programa

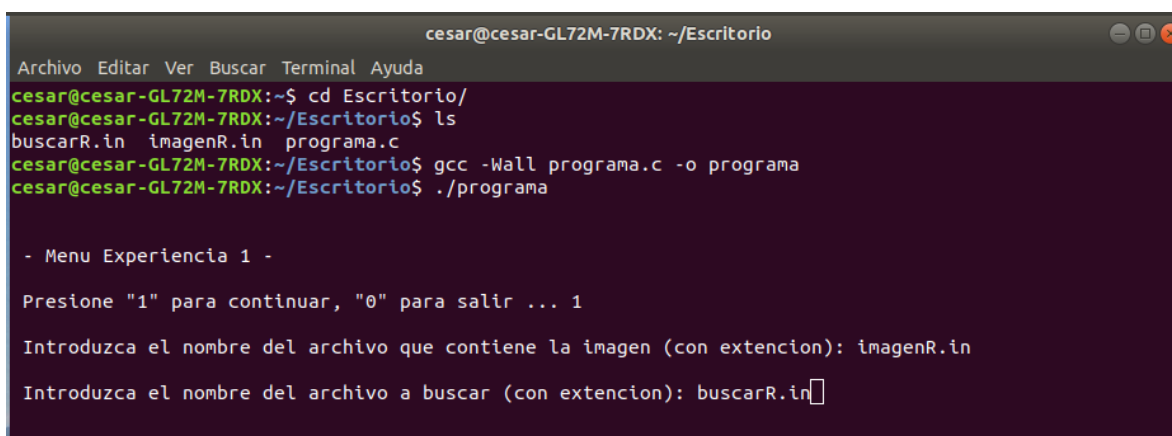
- Menu Experiencia 1 -

Presione "1" para continuar, "0" para salir ... 1

Introduzca el nombre del archivo que contiene la imagen (con extencion): imagenR.in

```

*Ilustración 11 Introducir nombre Imagen.*



```

cesar@cesar-GL72M-7RDX: ~/Escritorio
Archivo Editar Ver Buscar Terminal Ayuda
cesar@cesar-GL72M-7RDX:~$ cd Escritorio/
cesar@cesar-GL72M-7RDX:~/Escritorio$ ls
buscarR.in  imagenR.in  programa.c
cesar@cesar-GL72M-7RDX:~/Escritorio$ gcc -Wall programa.c -o programa
cesar@cesar-GL72M-7RDX:~/Escritorio$ ./programa

- Menu Experiencia 1 -

Presione "1" para continuar, "0" para salir ... 1

Introduzca el nombre del archivo que contiene la imagen (con extencion): imagenR.in

Introduzca el nombre del archivo a buscar (con extencion): buscarR.in

```

*Ilustración 12 Introducir nombre imagen a buscar.*

De esta forma se entregará el resultado final de la moda, media, mediana y las mayores y menores similitudes encontradas

### **3.3 CÓMO COMPILAR Y EJECUTAR EN WINDOWS**

Antes de compilar y ejecutar el programa en el ambiente Windows, es necesario que tanto el archivo que contiene la imagen (“imagen.in”), su imagen a buscar (“buscar.in”), como el archivo que contiene el programa (“programa.c”) se encuentren en el escritorio.

Luego, se procede a abrir la consola (cmd) de Windows. Estando ahí, para compilar es necesario acceder al escritorio, por lo que se debe escribir “cd Desktop” y presionar Enter.

Teniendo esto, ahora se procede a escribir el comando “-gcc programa.c -o programa.exe” y a continuación presionar la tecla Enter.

Finalizado esto, automáticamente se crea un nuevo archivo en el escritorio y así, el programa ha sido compilado.

Para ejecutar el programa sólo es necesario escribir a continuación “programa.exe” y presionar la tecla Enter. En la misma consola se entregará el determinante deseado.

### **3.4 FUNCIONALIDADES DEL PROGRAMA**

El presente programa, puede encontrar una imagen dentro de otra imagen, siempre y cuando se entregue un archivo con la imagen, un archivo con la imagen a buscar que cumplan con el siguiente formato (Véase Ilustración 1 y 2 respectivamente).

### **3.5 POSIBLES ERRORES**

Para la correcta ejecución del algoritmo, es necesario que el archivo que contiene la imagen y la imagen a buscar se encuentre en la misma carpeta que contiene el algoritmo, de no cumplirse esto al ejecutarse el programa, se entregará un error y no podrá buscarse sus coincidencias.

Por otro lado, el programa tampoco funcionará, en caso de que el archivo que contiene la imagen y la imagen a buscar no esté correctamente definido, es decir, que este no tenga el formato explicado anteriormente (Véase Ilustración 1 y 2 respectivamente).

Como el algoritmo desarrollado no es realmente eficiente, si se le entrega una matriz de un tamaño considerable, al ejecutarse el programa, este puede caerse o bien puede tomar demasiado tiempo en realizar las operaciones.