****

**TRABAJOS REALIZADOS EN EL PRIMER PARCIAL**

MATERIA: Estructura de Datos

NRC: 2967

**Introduccion**

A continuacion se presenta un informe detallado de los dieciocho programas elaborados a lo largo del primer parcial, sean deberes, trabajos en clase, pruebas o extras, los cuales constan de una definicion, objetivo, codigo y ejecucion (por medio de capturas).

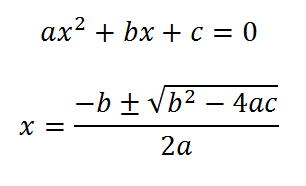
El codigo ha sido embellecido por medio de una herramienta de software llamda “Notepad++” la cual permite ingresar codigo en formato de programacion para un mejor entendimiento.

**Programas**

1. **Función Matemática (TDA)**

**Descripción**

La siguiente aplicación tiene como objetivo calcular las raíces de una ecuación de segundo grado usando un tipo de dato abstracto (TDA) para lograr eso se ocupará la fórmula general de una ecuación cuadrática, pidiendo al usuario los coeficientes de la ecuación a calcular.



**Objetivo de la aplicación**

Calcular las raíces de una ecuación de segundo grado mediante la fórmula general..

**Código de la aplicación**

* La clase Fórmula nos permite alojar los coeficientes de nuestra fórmula a desarrollar para luego ocuparlas en nuestra clase Raíz.

class Raiz**;**

class Formula

**{**

public**:**

Formula**();**

float getNum1**(**void**);**

void setNum1**(**float newNum1**);**

float getNum2**(**void**);**

void setNum2**(**float newNum2**);**

float getNum3**(**void**);**

void setNum3**(**float newNum3**);**

Raiz**\*** raiz**;**

protected**:**

private**:**

float num1**;**

float num2**;**

float num3**;**

**};**

#include "Raiz.h"

#include "Formula.h"

Formula**::**Formula**()**

**{**

**}**

float Formula**::**getNum1**(**void**)**

**{**

**return** num1**;**

**}**

void Formula**::**setNum1**(**float newNum1**)**

**{**

num1 **=** newNum1**;**

**}**

float Formula**::**getNum2**(**void**)**

**{**

**return** num2**;**

**}**

void Formula**::**setNum2**(**float newNum2**)**

**{**

num2 **=** newNum2**;**

**}**

float Formula**::**getNum3**(**void**)**

**{**

**return** num3**;**

**}**

void Formula**::**setNum3**(**float newNum3**)**

**{**

num3 **=** newNum3**;**

**}**

* La clase Raíz se encargará de calcular mediante la fórmula presentada anteriormente las raíces de nuestra ecuación

class Raiz

**{**

public**:**

void calcular**(**float a**,** float b**,** float c**);**

float getResult1**(**void**);**

void setResult1**(**float newResult1**);**

float getResult2**(**void**);**

void setResult2**(**float newResult2**);**

Raiz**();**

protected**:**

private**:**

float result1**;**

float result2**;**

**};**

#include<iostream>

#include<math.h>

#include "Raiz.h"

**using** **namespace** std**;**

void Raiz**::**calcular**(**float a**,** float b**,** float c**)**

**{**

Raiz raiz**;**

float raizValidacion**;**

raizValidacion**=**b**\***b**-**4**\***a**\***c**;**

**if(**raizValidacion**<**0**){**

cout**<<**"No existe solucion"**<<**endl**;**

**}else{**

raiz**.**setResult1**((-**b**+**sqrt**(**raizValidacion**))/(**2**\***a**));**

raiz**.**setResult2**((-**b**-**sqrt**(**raizValidacion**))/(**2**\***a**));**

cout**<<**"La primera solucion es: x1="**<<**raiz**.**getResult1**()<<**endl**;**

cout**<<**"La segunda solucion es: x2="**<<**raiz**.**getResult2**()<<**endl**;**

**}**

**}**

float Raiz**::**getResult1**(**void**)**

**{**

**return** result1**;**

**}**

void Raiz**::**setResult1**(**float newResult1**)**

**{**

result1 **=** newResult1**;**

**}**

float Raiz**::**getResult2**(**void**)**

**{**

**return** result2**;**

**}**

void Raiz**::**setResult2**(**float newResult2**)**

**{**

result2 **=** newResult2**;**

**}**

Raiz**::**Raiz**()**

**{**

**}**

* La librería “Ingreso.h” nos permitirá que el usuario ingrese los datos para nuestra aplicación

#include <iostream>

#include "Validacion.h"

**using** **namespace** std**;**

class Ingreso **{**

public**:**

string leer**(**string**,**int**);**

**};**

string Ingreso**::**leer**(**string mensage**,**int tipo**)** **{**

Validacion validacion**;**

string entrada**;**

cout **<<** mensage **<<** endl**;**

cin **>>** entrada**;**

**while** **(**validacion**.**validar**(**entrada**,** tipo**))** **{**

cout **<<** "Valor no valido reingrese" **<<** endl**;**

cin **>>** entrada**;**

**}**

**return** entrada**;**

**}**

* La librería “Validar.h” nos ayudará a verificar que los datos ingresados sean correctos..

#include <iostream>

#include <ctype.h>

#include <string.h>

**using** **namespace** std**;**

class Validacion **{**

public**:**

bool validar**(**string**,** int**);**

**};**

bool Validacion**::**validar**(**string entrada**,** int tipo**)** **{**

int contador **=** 0**;**

**try** **{**

**for** **(**int i **=** 0**;** i **<** entrada**.**length**();** i**++)** **{**

**if** **(**isalpha**(**entrada**[**i**]))** **{**

**throw** 1**;**

**}**

**if** **(!**isdigit**(**entrada**[**i**])** **&&** tipo **==** 1**)** **{**

**throw** 1**;**

**}**

**if** **(**entrada**[**i**]** **==** '.'**)** **{**

contador**++;**

**}**

**if** **((**isdigit**(**entrada**[**i**])** **==** 0 **&&** entrada**[**i**]** **!=** '.' **&&** entrada**[**i**]** **!=** '-'**)** **||** **(**contador**>**1**))** **{**

**throw** 1**;**

**}**

**}**

**}**

**catch** **(**int e**)** **{**

**return** **true;**

**}**

**return** **false;**

**}**

* Este será nuestro aplicativo donde se ejecutará nuestro programa.

#include<iostream>

#include <sstream>

#include"Formula.cpp"

#include"Raiz.cpp"

#include"Ingreso.h"

**using** **namespace** std**;**

int main**(){**

Formula obj**;**

Raiz obj2**;**

Ingreso ingreso**;**

float a**,**b**,**c**;**

string dim**;**

cout**<<**">Ingrese los datos (ax^2+bx+c)"**<<**endl**;**

dim**=**ingreso**.**leer**(**"Ingrese el coeficiente a"**,**2**);**

istringstream **(**dim**)** **>>** a**;**

obj**.**setNum1**(**a**);**

dim**=**ingreso**.**leer**(**"Ingrese el coeficiente b"**,**2**);**

istringstream **(**dim**)** **>>** b**;**

obj**.**setNum2**(**b**);**

dim**=**ingreso**.**leer**(**"Ingrese el coeficiente c"**,**2**);**

istringstream **(**dim**)** **>>** c**;**

obj**.**setNum3**(**c**);**

obj2**.**calcular**(**obj**.**getNum1**(),**obj**.**getNum2**(),**obj**.**getNum3**());**

**return** 0**;**

**}**

**Ejecución del aplicativo**

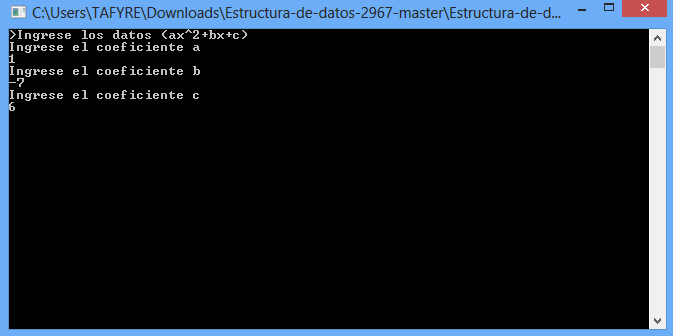


Gráfico 1. Ingreso coeficientes por parte del usuario.

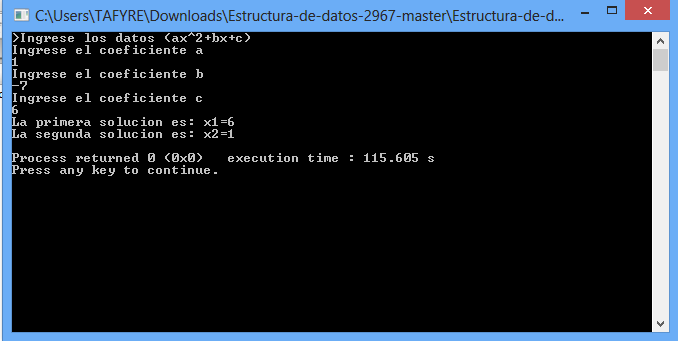


Gráfico 2. Impresión de resultados.

1. **Funciones Trigonométricas**

**Descripción**

La siguiente aplicación tiene como objetivo desarrollar las funciones trigonométricas se definen comúnmente como el cociente entre dos lados de un triángulo rectángulo, asociado a sus ángulos.

Las funciones trigonométricas son funciones cuyos valores son extensiones del concepto de razón trigonométrica en un triángulo rectángulo trazado en una circunferencia unitaria (de radio unidad).

**Objetivo de la aplicación**

Calcular las Funciones trigonométricas básicas (seno, coseno, tangente) para la aplicación de las mismas en operaciones matemáticas.

**Código de la aplicación**

* La librería “Funciones Trigonométricas” donde se encuentran las funciones que permiten calcular los valores de las funciones trigonométricas (seno, coseno, tangente).

#include <iostream>

#include <stdio.h>

#include <math.h>

#define PI 3.14159265359

**using** **namespace** std**;**

class FuncionesTrigonometricas

**{**

public**:**

int factorial**(**int angulo**);**

float seno**(**float angulo**);**

float coseno**(**float angulo**);**

float tangente**(**float angulo**);**

float radianes**(**float angulo**);**

int potencia**(**int base**,**int exponente**);**

protected**:**

private**:**

**};**

//Factorial de un Numero

int FuncionesTrigonometricas**::**factorial **(**int angulo**)**

**{** int factor**,**i**;**

factor**=**1**;**

**for** **(**i**=**1**;**i**<=**angulo**;**i**++)** **{**

factor**=**factor**\***i**;**

**}**

**return** **(**factor**);**

**}**

// Funcion Seno para un angulo radianes

float FuncionesTrigonometricas**::**seno**(**float angulo**)**

**{**

float resultado**;**

int i**,** posicion**,** precision**;**

resultado**=** angulo**;**

precision**=**5**;**

**for(**i**=**1**;** i**<=**precision**;** i**++){**

posicion **=** i **\*** 2 **+** 1**;**

**if(**i**%**2**==**0**)**

resultado **+=** potencia**(**angulo**,** posicion**)** **/** factorial**(**posicion**);**

**else**

resultado **-=** potencia**(**angulo**,** posicion**)** **/** factorial**(**posicion**);**

**}**

**return** resultado**;**

**}**

//Funcion Coseno para un angulo radianes

float FuncionesTrigonometricas**::**coseno**(**float angulo**){**

float resultado**;**

int i**,** posicion**,** precision**;**

resultado**=** 1**;**

precision**=**5**;**

**for(**i**=**1**;** i**<=**precision**;** i**++){**

posicion **=** i **\*** 2**;**

**if(**i**%**2**==**0**)**

resultado **+=** potencia**(**angulo**,** posicion**)** **/** factorial**(**posicion**);**

**else**

resultado **-=** potencia**(**angulo**,** posicion**)** **/** factorial**(**posicion**);**

**}**

**return** resultado**;**

**}**

//Funcion Tangente para un angulo radianes

float FuncionesTrigonometricas**::**tangente**(**float angulo**){**

float coseno**;**

float seno**;**

float resultado**;**

coseno**=**FuncionesTrigonometricas**::**coseno**(**angulo**);**

seno**=**FuncionesTrigonometricas**::**seno**(**angulo**);**

resultado**=**seno**/**coseno**;**

**return** resultado**;**

**}**

int FuncionesTrigonometricas**::**potencia**(**int base**,**int exponente**){**

int resultado **=** base**;**

**if(**exponente**>**0**){**

**return** resultado**\***potencia**(**base**,**exponente**-**1**);**

**}else{**

**return** 1**;**

**}**

**}**

float FuncionesTrigonometricas**::**radianes**(**float angulo**){**

float resultado**;**

resultado **=** angulo**\*(**PI**/**180**);**

**return** resultado**;**

**}**

* La librería “Ingreso.h” es la que permite que el usuario digite los datos que van a ser procesados para el cálculo de las funciones trigonométricas.

#include <iostream>

#include "Validacion.h"

**using** **namespace** std**;**

class Ingreso **{**

public**:**

string leer**(**string**,**int**);**

**};**

string Ingreso**::**leer**(**string mensage**,**int tipo**)** **{**

Validacion validacion**;**

string entrada**;**

cout **<<** mensage **<<** endl**;**

cin **>>** entrada**;**

**while** **(**validacion**.**validar**(**entrada**,** tipo**))** **{**

cout **<<** "Valor no valido reingrese" **<<** endl**;**

cin **>>** entrada**;**

**}**

**return** entrada**;**

**}**

* La librería “Validadcion.h” verifica que el dato ingresado por el usuario sea correcto.

#include <iostream>

#include <ctype.h>

#include <string.h>

**using** **namespace** std**;**

class Validacion **{**

public**:**

bool validar**(**string**,** int**);**

**};**

bool Validacion**::**validar**(**string entrada**,** int tipo**)** **{**

int contador **=** 0**;**

**try** **{**

**for** **(**int i **=** 0**;** i **<** entrada**.**length**();** i**++)** **{**

**if** **(**isalpha**(**entrada**[**i**]))** **{**

**throw** 1**;**

**}**

**if** **(!**isdigit**(**entrada**[**i**])** **&&** tipo **==** 1**)** **{**

**throw** 1**;**

**}**

**if** **(**entrada**[**i**]** **==** '.'**)** **{**

contador**++;**

**}**

**if** **((**isdigit**(**entrada**[**i**])** **==** 0 **&&** entrada**[**i**]** **!=** '.'**)** **||** **(**contador**>**1**))** **{**

**throw** 1**;**

**}**

**}**

**}**

**catch** **(**int e**)** **{**

**return** **true;**

**}**

**return** **false;**

**}**

* Este será nuestro aplicativo donde se ejecutará nuestro programa.

#include <iostream>

#include <stdlib.h>

#include "FuncionesTrigonometricas.h"

#include "Ingreso.h"

**using** **namespace** std**;**

int main**(){**

FuncionesTrigonometricas funciones**;**

Ingreso ingreso**;**

string angulo1**;**

float angulo2**;**

float calculo1**;**

float calculo2**;**

float calculo3**;**

cout**<<** "\*\*\*CALCULO DE FUNCIONES TRIGONOMETRICAS : SENO, COSENO, TANGENTE\*\*\*"**<<**endl**<<**endl**;**

angulo1 **=** ingreso**.**leer**(**"Ingrese un angulo: "**,**2**);**

angulo2 **=** funciones**.**radianes**(**atof**(**angulo1**.**c\_str**()));**

calculo1 **=**funciones**.**seno**(**angulo2**);**

calculo2 **=**funciones**.**coseno**(**angulo2**);**

calculo3 **=**funciones**.**tangente**(**angulo2**);**

system**(**"cls"**);**

cout**<<** "\*\*\*CALCULO DE FUNCIONES TRIGONOMETRICAS : SENO, COSENO, TANGENTE\*\*\*"**<<**endl**<<**endl**;**

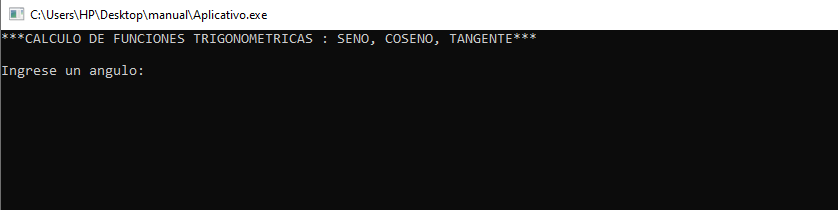
cout**<<** endl **<<** "SENO DE: "**<<** angulo1 **<<** " ES= " **<<** calculo1 **<<**endl**;**

cout**<<** "COSENO DE: "**<<** angulo1 **<<** " ES= " **<<**calculo2 **<<**endl**;**

cout**<<** "TANGENTE DE: "**<<** angulo1 **<<** " ES= " **<<**calculo3 **<<**endl**;**

**}**

**Ejecución del Aplicativo**

Figura 1 : Ingreso de datos por parte del usuario

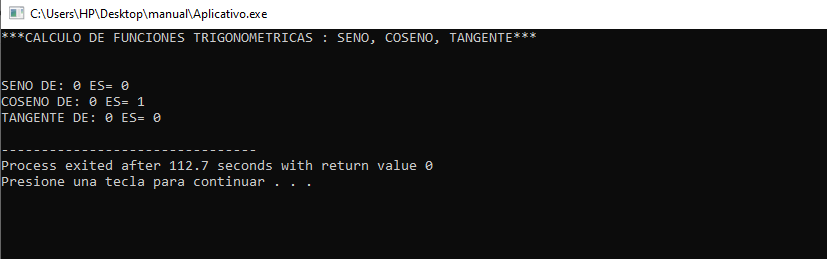


Figura 2: Funciones trigonométricas

1. **Sudoku**

**Descripción**

El objetivo del sudoku es rellenar una cuadrícula de 9 × 9 celdas (81 casillas) dividida en sub cuadrículas de 3 × 3 (también llamadas "cajas" o "regiones") con las cifras del 1 al 9 partiendo de algunos números ya dispuestos en algunas de las celdas. Aunque se podrían usar colores, letras, figuras, se conviene en usar números para mayor claridad, lo que importa, es que sean nueve elementos diferenciados, que no se deben repetir en una misma fila, columna o sub cuadrícula.

**Objetivo de la aplicación**

Simular el juego Sudoku de tal manera que el aplicativo siempre dé como resultado un juego correcto siguiendo las reglas del mismo.

**Código de la aplicación**

* En la clase “Sudoku.cpp” se encuentran todas las funciones necesarias para la resolución del juego en las cuales se encuentran todas las restricciones del mismo.

#include "Sudoku.h"

int**\*\*** Sudoku**::**inicializar**(**int tamanio**)** **{**

matriz **=** **(**int**\*\*)** malloc**(sizeof** **(**int **\*)\***tamanio**);**

**for** **(**int i **=** 0**;** i **<** tamanio**;** i**++)** **{**

**\*(**matriz **+** i**)** **=** **(**int**\*)** malloc**(sizeof** **(**int**\*)\***tamanio**);**

**}**

**return** matriz**;**

**}**

void Sudoku**::**tablero**(**int **\*\***matriz**)** **{**

cout **<<** "-------------SUDOKU---------------\n"**;**

**for** **(**int i **=** 0**;** i **<** 9**;** i**++)** **{**

printf**(**"\n"**);**

**if** **(**i **==** 3 **||** i **==** 6**)** **{**

cout **<<** "----------------------------------\n"**;**

**}**

**for** **(**int j **=** 0**;** j **<** 9**;** j**++)** **{**

**if** **(**j **==** 3 **||** j **==** 6**)** **{**

printf**(**" |"**);**

**}**

cout **<<** " " **<<** **\*(\*(**matriz **+** i**)** **+** j**)** **<<** " "**;**

**}**

**}**

**}**

int**\*\*** Sudoku**::**hacerSolucion**(**int **\*\***matriz**)** **{**

srand**(**time**(NULL));**

**do** **{**

contador2 **=** 0**;**

matriz **=** inicializar**(**10**);**

//Enceramos la matriz

**for** **(**int i **=** 0**;** i **<** 9**;** i**++)** **{**

**for** **(**int j **=** 0**;** j **<** 9**;** j**++)** **{**

**\*(\*(**matriz **+** i**)** **+** j**)** **=** 0**;**

contador1 **=** 0**;**

**do** **{**

//Se almacenan numeros aleatorios del 1 al 9

aleatorio **=** rand**()** **%** 9 **+** 1**;**

//Si la funcion controlar devuelve 1 se almacena el numero aleatorio generado, si devuelve 0 vuelve a generar un numero aleatorio

detente **=** controlar**(**aleatorio**,** 9**,** i**,** j**,** matriz**);**

contador1**++;**

//Hace que termine los dos ciclos for

**if** **(**contador1 **>** 15**)** **{**

i **=** 9**;**

j **=** 9**;**

**}**

**}** **while** **(**detente **==** 1**);**

**\*(\*(**matriz **+** i**)** **+** j**)** **=** aleatorio**;**

//Aumenta el numero de 1 en 1 hasta llegar a 81 para terminar el ciclo do-while

contador2**++;**

**}**

**}**

**}** **while** **(**contador2 **!=** 81**);**

**return** matriz**;**

**}**

int Sudoku**::**controlar**(**int numero**,** int tamanio**,** int contador1**,** int contador2**,** int **\*\***matriz**)** **{**

detente **=** 0**;**

**for** **(**int i **=** 0**;** **(**i **<** tamanio **&&** detente **==** 0**);** i**++)** **{**

//Se va desplazando y comprueba si el espacio en donde está ubicado es igual al numero aleatorio generado tanto en fila como columna

**if** **(\*(\*(**matriz **+** contador1**)** **+** i**)** **==** numero **||** **\*(\*(**matriz **+** i**)** **+** contador2**)** **==** numero**)** **{**

detente **=** 1**;**

**}**

**}**

**return** detente**;**

**}**

* En la librería “Sudoku.h” se encuentra la funciones declaradas y las variables que serán utilizadas en el desarrollo de funciones para la aplicación.

#include <iostream>

#include <time.h>

#include <stdlib.h>

**using** **namespace** std**;**

class Sudoku **{**

private**:**

int **\*\***matriz**;**

int detente**;**

int aleatorio**;**

int contador1**;**

int contador2**;**

public**:**

int**\*\*** inicializar**(**int**);**

void tablero**(**int**\*\*);**

int**\*\*** hacerSolucion**(**int**\*\*);**

int controlar**(**int**,** int**,** int**,** int**,** int**\*\*);**

**};**

* Este será nuestro aplicativo donde se ejecutará nuestro programa.

#include "Sudoku.cpp"

int main**()** **{**

Sudoku sudoku**;**

int **\*\***matriz**;**

matriz **=** sudoku**.**hacerSolucion**(**matriz**);**

sudoku**.**tablero**(**matriz**);**

**return** 0**;**

**}**

**Ejecución del Aplicativo**

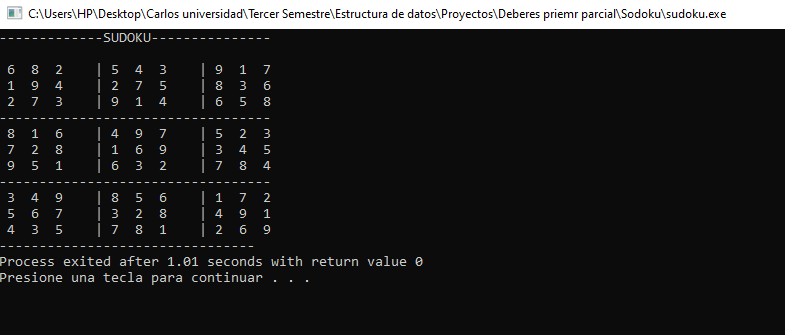


Figura 1 Tablero de solución Sudoku

1. **Matriz Identidad**

**Descripción**

Es una matriz que cumple la propiedad de ser el elemento neutro del producto de matrices. Esto quiere decir que el producto de cualquier matriz por la matriz identidad (donde dicho producto esté definido) no tiene ningún efecto.

Para obtener la matriz identidad se debe multiplicar la matriz con su inversa.

**Objetivo de la aplicación**

Calcular la matriz Identidad de cualquier matriz ingresada de dimensión n, mediante el uso de procesos matemáticos.

**Código de la aplicación**

* A continuación presentamos el código de la aplicación “identidad.h”

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

class Identidad **{**

private**:**

float **\*\***matriz**;**

float **\***matriz1**;**

float elemento**;**

float coeficiente**;**

float **\***vector**;**

float **\*\***respuesta**;**

public**:**

float**\*** inicializar1**(**int**);**

float**\*\*** inicializar**(**int**);**

float**\*\*** encerar**(**int**,**float**\*\*);**

float**\*** encerar1**(**int**,**float**\*);**

float**\*\*** ingresarNumeros**(**int**,**float**\*\*);**

float**\*\*** calcularInversa**(**int**,**float**\*\*);**

float**\*\*** multiplicar**(**int**,**float**\*\*,**float**\*\*);**

void mostrarInversa**(**int**,**float**\*\*);**

void mostrar**(**int**,**float**\*\*);**

**};**

float**\*** Identidad**::**inicializar1**(**int tamanio**)** **{**

matriz1 **=** **(**float**\*)**malloc**(**tamanio **\*sizeof(**float**));**

**return** matriz1**;**

**}**

float**\*** Identidad**::**encerar1**(**int tamanio**,**float **\***matriz**)** **{**

**for(**int i**=**0**;** i**<**tamanio**;** i**++)** **{**

**for(**int j**=**0**;** j**<**tamanio**;** j**++)** **{**

**\*(**matriz**+**i**)** **=** 0**;**

**}**

**}**

**return** matriz**;**

**}**

float**\*\*** Identidad**::**inicializar**(**int tamanio**)** **{**

matriz **=** **(**float**\*\*)**malloc**(sizeof(**float **\*)\***tamanio**);**

**for(**int i**=**0**;** i**<**tamanio**;** i**++)** **{**

**\*(**matriz**+**i**)** **=** **(**float**\*)**malloc**(sizeof(**float**\*)\***tamanio**);**

**}**

**return** matriz**;**

**}**

float**\*\*** Identidad**::**encerar**(**int tamanio**,**float **\*\***matriz**)** **{**

**for(**int i**=**0**;** i**<**tamanio**;** i**++)** **{**

**for(**int j**=**0**;** j**<**tamanio**;** j**++)** **{**

**\*(\*(**matriz**+**i**)+**j**)** **=** 0**;**

**}**

**}**

**return** matriz**;**

**}**

float**\*\*** Identidad**::**ingresarNumeros**(**int tamanio**,**float **\*\***matriz**)** **{**

matriz **=** inicializar**(**tamanio**);**

matriz **=** encerar**(**tamanio**,**matriz**);**

**for(**int i**=**0**;**i**<**tamanio**;**i**++)** **{**

**for(**int j**=**0**;**j**<**tamanio**;**j**++)** **{**

scanf**(**"%f"**,\*(**matriz**+**i**)+**j**);**

**}**

**}**

**return** matriz**;**

**}**

float**\*\*** Identidad**::**calcularInversa**(**int tamanio**,**float **\*\***matriz**)** **{**

vector **=** inicializar1**(**tamanio**);**

vector **=** encerar1**(**tamanio**,**vector**);**

**for(**int i**=**0**;** i**<**tamanio**;** i**++)**

**for(**int j**=**tamanio**;** j**<**2**\***tamanio**;** j**++)** **{**

**if(**i**==(**j**-**tamanio**))** **{**

**\*(\*(**matriz**+**i**)+**j**)** **=** 1**;**

**}else** **{**

**\*(\*(**matriz**+**i**)+**j**)** **=** 0**;**

**}**

**}**

**for(**int s**=**0**;** s**<**tamanio**;** s**++)** **{**

elemento **=** **\*(\*(**matriz**+**s**)+**s**);**

**for(**int j**=**0**;** j**<**2**\***tamanio**;** j**++)** **{**

**\*(\*(**matriz**+**s**)+**j**)** **=** **\*(\*(**matriz**+**s**)+**j**)** **/** elemento**;**

**}**

**for(**int i**=**0**;** i**<**tamanio**;** i**++)** **{**

**if(**i**==**s**)** **{**

**;**

**}**

**else** **{**

coeficiente **=** **\*(\*(**matriz**+**i**)+**s**);**

**for(**int j**=**0**;** j**<**2**\***tamanio**;** j**++)**

**\*(**vector**+**j**)** **=** **\*(\*(**matriz**+**s**)+**j**)** **\*** **(**coeficiente**\*-**1**);**

**for(**int j**=**0**;** j**<**2**\***tamanio**;** j**++)**

**\*(\*(**matriz**+**i**)+**j**)** **=** **\*(\*(**matriz**+**i**)+**j**)** **+** **\*(**vector**+**j**);**

**}**

**}**

**}**

**return** matriz**;**

**}**

void Identidad**::**mostrar**(**int tamanio**,** float **\*\***matriz**)** **{**

**for(**int i**=**0**;** i**<**tamanio**;** i**++)** **{**

**for(**int j**=**0**;** j**<**tamanio**;** j**++)** **{**

printf**(**" %0.0f "**,\*(\*(**matriz**+**i**)+**j**));**

**}**

printf**(**"\n"**);**

**}**

**}**

void Identidad**::**mostrarInversa**(**int tamanio**,** float **\*\***matriz**)** **{**

**for(**int i**=**0**;** i**<**tamanio**;** i**++)** **{**

**for(**int j**=**tamanio**;** j**<**2**\***tamanio**;** j**++)** **{**

printf**(**" %0.1f "**,\*(\*(**matriz**+**i**)+**j**));**

**}**

printf**(**"\n"**);**

**}**

**}**

float**\*\*** Identidad**::**multiplicar**(**int tamanio**,**float **\*\***matriz1**,** float **\*\***matriz2**)** **{**

respuesta **=** inicializar**(**tamanio**);**

respuesta **=** encerar**(**tamanio**,**respuesta**);**

**for(**int i**=**0**;**i**<**tamanio**;**i**++)** **{**

**for(**int j**=**0**;**j**<**tamanio**;**j**++)** **{**

**for(**int k**=**0**;**k**<**tamanio**;**k**++)** **{**

**\*(\*(**respuesta**+**i**)+**j**)** **=** **\*(\*(**respuesta**+**i**)+**j**)** **+** **(\*(\*(**matriz1**+**i**)+**k**))** **\*** **(\*(\*(**matriz2**+**k**)+**j**));**

**}**

**}**

**}**

**return** respuesta**;**

**}**

* Luego tenemos la clase ingreso la cual valida nuestras entradas hacia el programa “ingreso.h”

#include <stdio.h>

#include <iostream>

#include <string.h>

#include <sstream>

#include <stdlib.h>

**using** **namespace** std**;**

class Ingreso**{**

public**:**

char**\*** ingresar**(**char**\*);**

string ingresarSoloTexto**(**char**\*);**

int ingresarEntero**(**char **\*);**

float ingresarFlotante**(**char **\*);**

**};**

char**\*** Ingreso**::**ingresar**(**char**\*** msg**){**

char**\*** texto**;**

cout**<<**msg**<<**endl**;**

cin**>>**texto**;**

**return** texto**;**

**}**

string Ingreso**::**ingresarSoloTexto**(**char **\***msg**){**

float valor**;**

string texto**;**

string res**;**

**while** **(**1**)**

**{**

bool is\_valid **=** **true;**

cout **<<** msg **<<** endl**;**

getline**(**cin**,**texto**);**

**try{**

**for** **(**size\_t i **=** 0**;** i **<** texto**.**length**();** i**++)** **{**

**if** **(!**isalpha**(**texto**[**i**]))** **{**

**if(**texto**[**i**]** **==** ' '**){**

**continue;**

**}else{**

cout **<<** "Se debe ingresar solo letras\n"**;**

is\_valid **=** **false;**

**break;**

**}**

**}**

**}**

**}catch(**exception e**){**

cout**<<**"error"**;**

**}**

**if** **(**is\_valid**){**

res **=** texto**.**c\_str**();**

**break;**

**}**

**}**

**return** res**;**

**}**

float Ingreso**::**ingresarFlotante**(**char **\***msg**){**

float valor**;**

string numero**;**

**while** **(**1**)**

**{**

bool is\_valid **=** **true;**

cout **<<** msg **<<** endl**;**

cin **>>** numero**;**

**try{**

**for** **(**size\_t i **=** 0**;** i **<** numero**.**length**();** i**++)** **{**

**if** **(!**isdigit**(**numero**[**i**]))** **{**

**if(!(**numero**[**i**]==**'.'**)){**

cout **<<** "Se debe ingresar numeros\n"**;**

is\_valid **=** **false;**

**break;**

**}**

**}**

**}**

**}catch(**exception e**){**

cout**<<**"error"**;**

**}**

**if** **(**is\_valid**){**

stringstream geek**(**numero**);**

geek**>>**valor**;**

**break;**

**}**

**}**

**return** valor**;**

**}**

int Ingreso**::**ingresarEntero**(**char **\***msg**){**

int valor**;**

string numero**;**

char **\***res**;**

**while** **(**1**)**

**{**

bool is\_valid **=** **true;**

cout **<<** msg **<<** endl**;**

cin **>>** numero**;**

**try{**

**for** **(**size\_t i **=** 0**;** i **<** numero**.**length**();** i**++)** **{**

**if** **(!**isdigit**(**numero**[**i**]))** **{**

cout **<<** "Se debe ingresar numeros\n"**;**

is\_valid **=** **false;**

**break;**

**}**

**}**

**}catch(**exception e**){**

cout**<<**"error"**;**

**}**

**if** **(**is\_valid**){**

res **=** **(**char **\*)**numero**.**c\_str**();**

valor**=**atoi**(**res**);**

**break;**

**}**

**}**

**return** valor**;**

**}**

* Finalmente tenemos la implementación del método principal (Main) “main.cpp”

#include <iostream>

#include "identidad.h"

#include "ingreso.h"

**using** **namespace** std**;**

int main**(){**

Ingreso lee**;**

Identidad obj**;**

float **\*\***matriz**;**

float **\*\***inversa**;**

float **\***matriz1**;**

float elemento**;**

float coeficiente**;**

float **\***vector**;**

float **\*\***respuesta**;**

int dimension**=**0**;**

**do{**

system**(**"cls"**);**

dimension **=** lee**.**ingresarEntero**(**"Ingrese dimension de la matriz: "**);**

**}while(**dimension **==** 0 **||** dimension **<** 0**);**

//RESERVO MEMORIA PARA LAS MATRICES

matriz **=** obj**.**inicializar**(**dimension**);**

inversa **=** obj**.**inicializar**(**dimension**);**

respuesta **=** obj**.**inicializar**(**dimension**);**

//ENCERO LAS MATRICES PARA EVITAR CUALQUIER FALLA

obj**.**encerar**(**dimension**,**matriz**);**

obj**.**encerar**(**dimension**,**respuesta**);**

obj**.**encerar**(**dimension**,**inversa**);**

//PIDO QUE SE INGRESEN LOS VALORES DE LA MATRIZ

matriz **=** obj**.**ingresarNumeros**(**dimension**,**matriz**);**

cout**<<**"Matriz ingresada: "**<<**endl**;**

obj**.**mostrar**(**dimension**,**matriz**);**

cout**<<**"\n\n\n"**;**

//CALCULO LA INVERSA DE LA MATRIZ ANTES INGRESADA

inversa **=** obj**.**calcularInversa**(**dimension**,**matriz**);**

cout**<<**"Matriz inversa: "**<<**endl**;**

obj**.**mostrarInversa**(**dimension**,**inversa**);**

cout**<<**"\n\n\n"**;**

//MULTIPLICO LA MATRIZ CON SU INVERSA Y GUARDO EL RESULTADO (MATRIZ IDENTIDAD) EN RESPUESTA

respuesta **=** obj**.**multiplicar**(**dimension**,**matriz**,**inversa**);**

cout**<<**"Matriz resultante (identidad): "**<<**endl**;**

obj**.**mostrar**(**dimension**,**respuesta**);**

system**(**"pause"**);**

**return** 0**;**

**}**

**Ejecución de la aplicación**

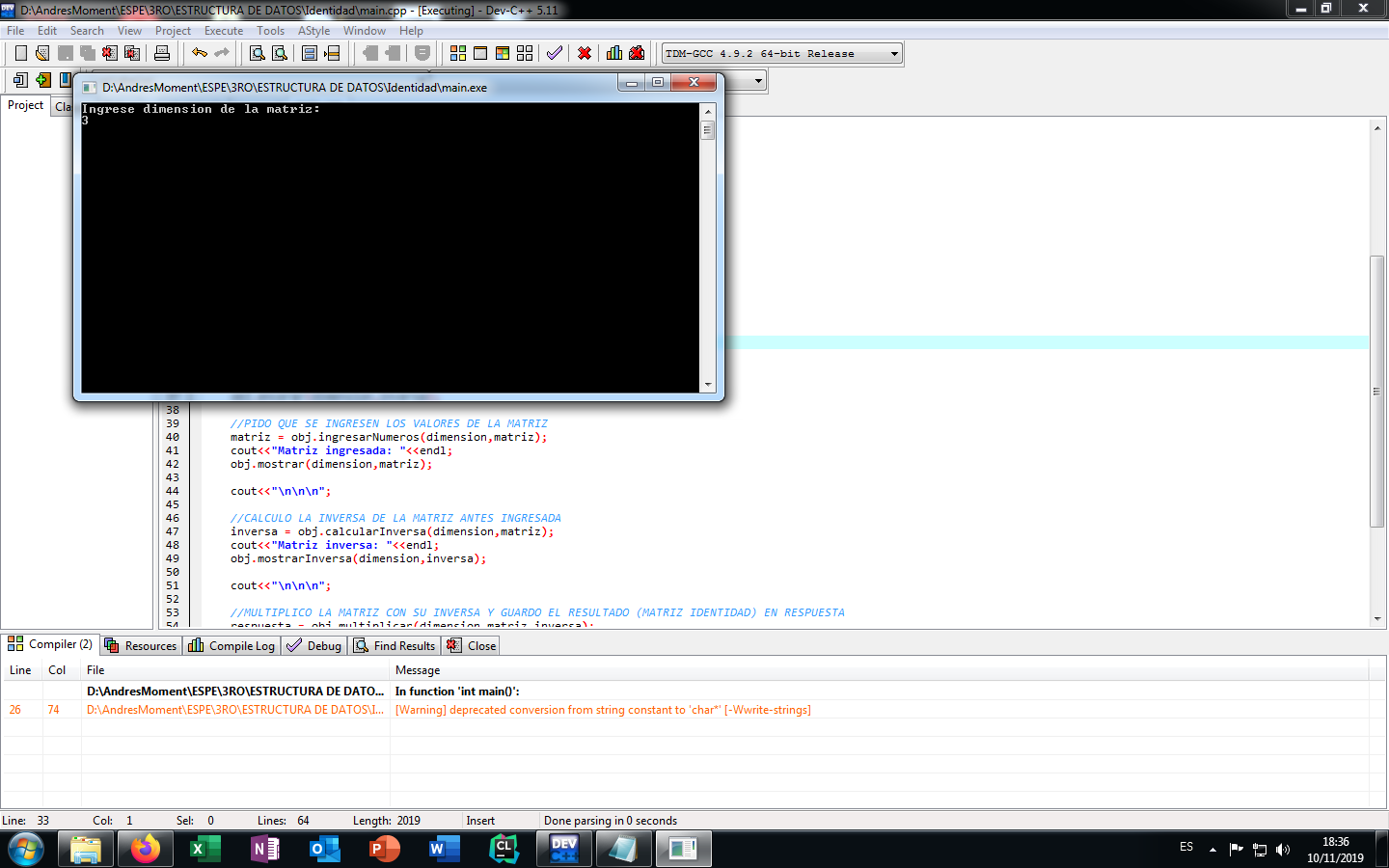
****

Figura 1: Ejecución de la aplicación

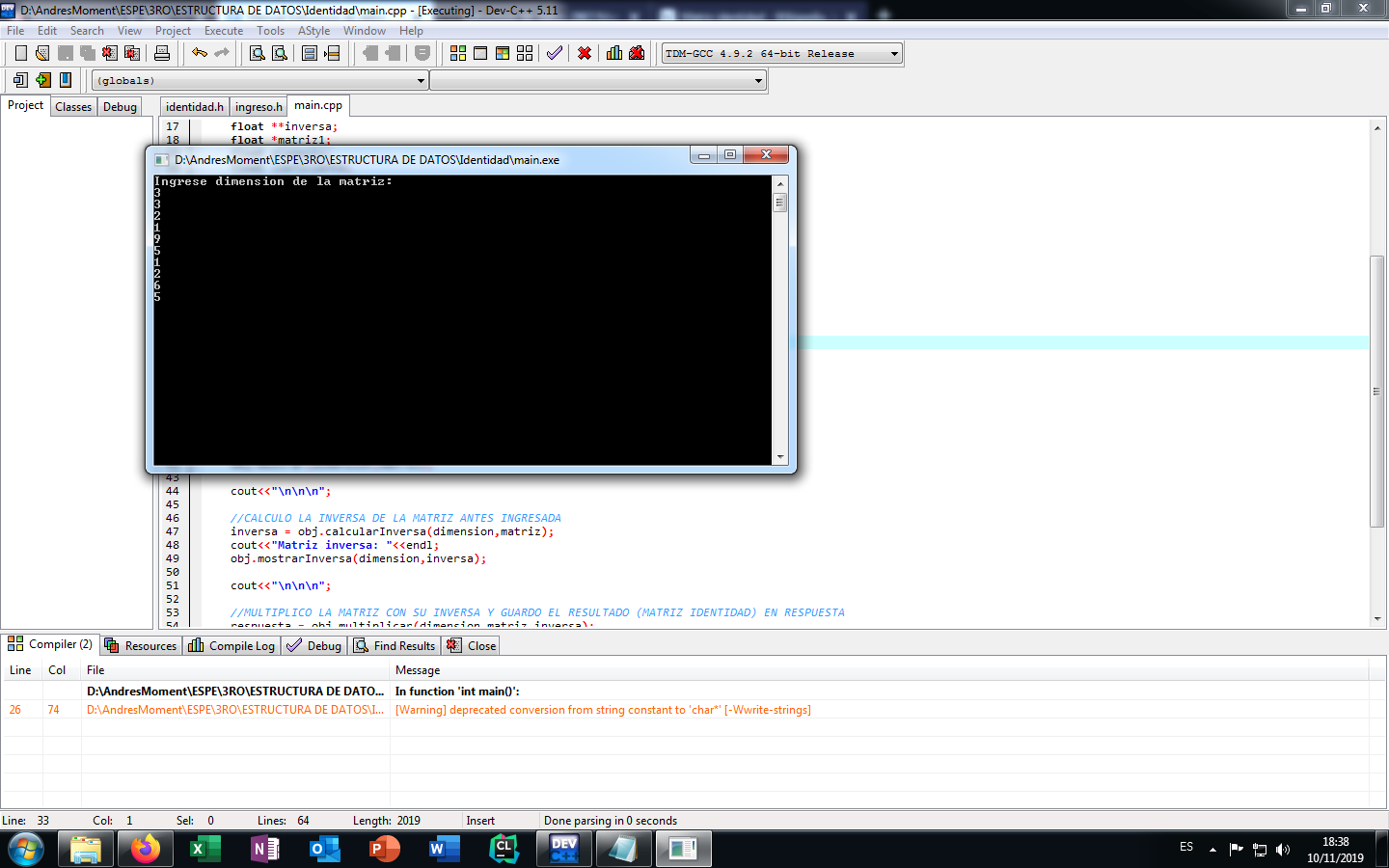


Figura 2: Ingreso de Datos

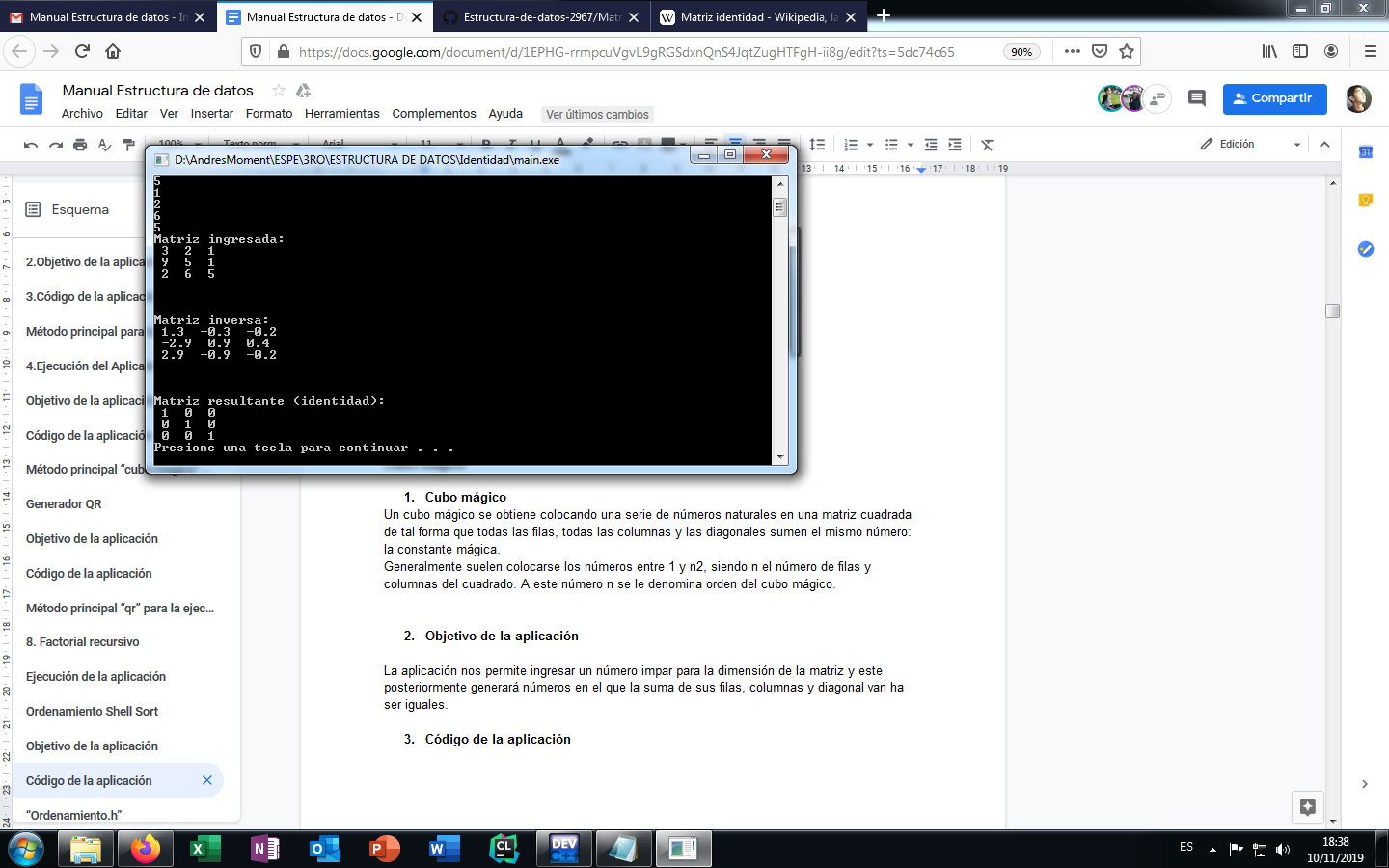


Figura 3: Cálculo de la Matriz identidad

1. **Cubo mágico**

**Descripción**

Un cubo mágico se obtiene colocando una serie de números naturales en una matriz cuadrada de tal forma que todas las filas, todas las columnas y las diagonales sumen el mismo número: la constante mágica.

Generalmente suelen colocarse los números entre 1 y n2, siendo n el número de filas y columnas del cuadrado. A este número n se le denomina orden del cubo mágico.

**Objetivo de la aplicación**

La aplicación nos permite ingresar un número impar para la dimensión de la matriz y este posteriormente generará números en el que la suma de sus filas, columnas y diagonal van ha ser iguales.

**Código de la aplicación**

* La librería “CuboMagico” nos permite encontrar los valores de filas y columnas que cumplan con la condición de que la suma de ellas y su diagonal sean iguales.

#include <iostream>

#include <stdlib.h>

#include <time.h>

#include <stdio.h>

**using** **namespace** std**;**

class CuboMagico**{**

private**:**

int **\*\***cuadrado**;**

public**:**

CuboMagico**(**int **\*\*);**

void llenar**(**int**,**int**);**

void mostrar**(**int**,** int**);**

int **\*\***getCuboMagicoCuadrado**();**

void setCuboMagicoCuadrado**(**int **\*\*);**

**};**

CuboMagico**::**CuboMagico**(**int **\*\***\_cuadrado**){**

cuadrado**=**\_cuadrado**;**

**}**

int **\*\***CuboMagico**::**getCuboMagicoCuadrado**(){**

**return** cuadrado**;**

**}**

void CuboMagico**::**setCuboMagicoCuadrado**(**int **\*\***\_cuadrado**){**

cuadrado**=**\_cuadrado**;**

**}**

void CuboMagico**::**llenar**(**int a**,**int b**){**

int x**=**0**,**k**=**0**,**p**=**1**,**j**=**0**,**t**=**0**,**s**=**0**,**d**=**0**;**

x**=(**b**-**1**)/**2**;**

t**=((**a**+**1**)/**2**);**

s**=(**a**-**x**-**1**)/**2**;**

d**=**x**;**

**for(**int g**=**0**;** g**<((**a**+**1**)/**2**);**g**++){**

j**=**g**;**

k**=**g**;**

**for(** int i**=**x**;**i**>=**j**;**i**--){**

**\*(\*(**cuadrado**+**i**)+**k**)=**p**;**

**if(** k**<(**d**-**s**)** **){**

**\*(\*(**cuadrado**+**i**)+(**k**+**t**))=**p**;**

**}**

**if(**k**>(**d**+**s**)){**

**\*(\*(**cuadrado**+**i**)+(**k**-**t**))=**p**;**

**}**

**if(**i**>(**d**+**s**)){**

**\*(\*(**cuadrado**+(**i**-**t**))+**k**)=**p**;**

**}**

**if(**i**<(**d**-**s**)){**

**\*(\*(**cuadrado**+(**i**+**t**))+**k**)=**p**;**

**}**

k**++;**

p**++;**

**}**

x**++;**

**}**

**}**

void CuboMagico**::**mostrar**(**int a**,**int b**){**

system**(**"cls"**);**

int x**=**0**,**t**=**0**,**s**=**0**;**

t**=((**a**+**1**)/**2**);**

x**=(**b**-**1**)/**2**;**

s**=(**a**-**x**-**1**)/**2**;**

**for(**int l**=**0**;**l**<**a**;**l**++){**

cout**<<**"\n\t"**;**

**for(**int h**=**0**;**h**<**b**;**h**++){**

**if(**h**>=(**x**-**s**)** **&&** h**<=(**x**+**s**)** **&&** l**<=(**x**+**s**)** **&&** l**>=(**x**-**s**)){**

cout**<<**"\t"**<<\*(\*(**cuadrado**+**l**)+**h**);**

**}**

**}**

**}**

**}**

* La librería “Ingreso.h” es la que permite al usuario dar el valor de las dimensiones del cubo mágico

#include <iostream>

#include "Validacion.h"

**using** **namespace** std**;**

class Ingreso **{**

public**:**

string leer**(**string**,**int**);**

**};**

string Ingreso**::**leer**(**string mensage**,**int tipo**)** **{**

Validacion validacion**;**

string entrada**;**

cout **<<** mensage **<<** endl**;**

cin **>>** entrada**;**

**while** **(**validacion**.**validar**(**entrada**,** tipo**))** **{**

cout **<<** "Valor no valido reingrese" **<<** endl**;**

cin **>>** entrada**;**

**}**

**return** entrada**;**

**}**

* La libreria “Validacion.h” en la que verifica que el valor ingresado sea correcto y cumpla con la condición de que solo puede ingresar un número

#include <iostream>

#include <ctype.h>

#include <string.h>

**using** **namespace** std**;**

class Validacion **{**

public**:**

bool validar**(**string**,** int**);**

**};**

bool Validacion**::**validar**(**string entrada**,** int tipo**)** **{**

int contador **=** 0**;**

**try** **{**

**for** **(**int i **=** 0**;** i **<** entrada**.**length**();** i**++)** **{**

**if** **(**isalpha**(**entrada**[**i**]))** **{**

**throw** 1**;**

**}**

**if** **(!**isdigit**(**entrada**[**i**])** **&&** tipo **==** 1**)** **{**

**throw** 1**;**

**}**

**if** **(**entrada**[**i**]** **==** '.'**)** **{**

contador**++;**

**}**

**if** **((**isdigit**(**entrada**[**i**])** **==** 0 **&&** entrada**[**i**]** **!=** '.'**)** **||** **(**contador**>**1**))** **{**

**throw** 1**;**

**}**

**}**

**}**

**catch** **(**int e**)** **{**

**return** **true;**

**}**

**return** **false;**

**}**

* Este será nuestro aplicativo donde se ejecutará nuestro programa.

#include <iostream>

#include <stdlib.h>

#include <time.h>

#include <stdio.h>

#include <sstream>

#include "CuboMagico.h"

#include "Ingreso.h"

**using** **namespace** std**;**

int main**()**

**{**

int **\*\***cuadrado**;**

Ingreso ingreso**;**

int num**;**

string dim**;**

**do{**

dim**=**ingreso**.**leer**(**"Digite el tamanio del arreglo, solo numeros impares: "**,**2**);**

istringstream **(**dim**)** **>>** num**;**

**}while(**num**%**2**==**0**||**num**<**0**);**

num**=**2**\***num**-**1**;**

cuadrado**=(**int**\*\*)**calloc**(**num**,sizeof(**int**\*));**

**for(**int i**=**0**;**i**<**num**;**i**++){**

**\*(**cuadrado**+**i**)=(**int**\*)**calloc**(**num**,sizeof(**int**));**

**}**

CuboMagico cubo**=**CuboMagico**(**cuadrado**);**

cubo**.**llenar**(**num**,**num**);**

cubo**.**mostrar**(**num**,**num**);**

system**(**"pause"**);**

**return** 0**;**

**}**

**Ejecución del aplicativo**

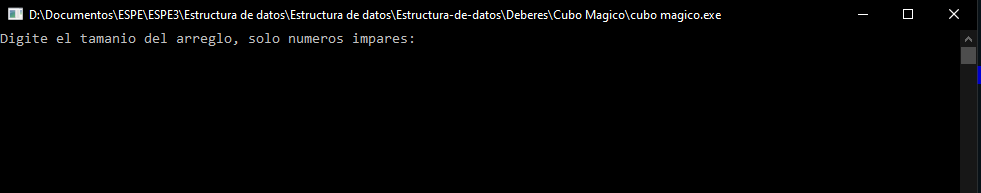
****

Figura 1. Ejecución de la aplicación

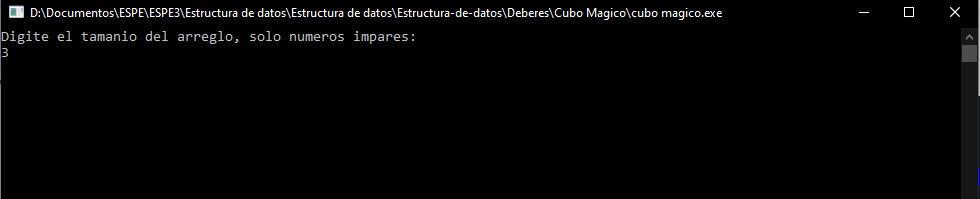
****

Figura 2. Ingreso de datos

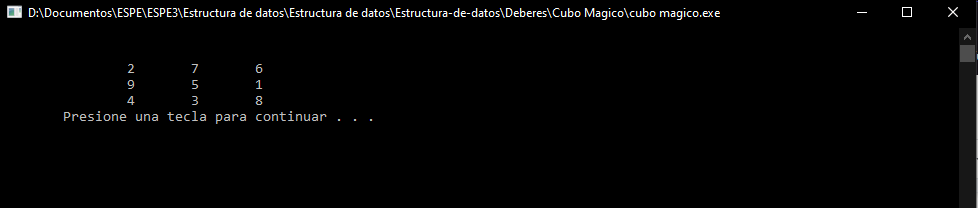


Figura 3. Cubo mágico

1. **Generador QR**

Un código QR es un código de barras bidimensional cuadrada que puede almacenar los datos codificados. La mayoría del tiempo los datos es un enlace a un sitio web. Un código QR es la evolución del código de barras. Es un módulo para almacenar información en una matriz de puntos o en un código de barras bidimensional.

**Objetivo de la aplicación**

Ingresar el tamaño que deseemos para el QR y este nos genera automáticamente una representación gráfica del código QR.

**Código de la aplicación**

* La librería “Ingreso.h” es la que permite al usuario dar el valor de las dimensiones del cubo mágico.

#include <iostream>

#include <stdlib.h>

#include <string.h>

**using** **namespace** std**;**

class Ingreso **{**

public**:**

int ingresarEntero**();**

double ingresarDouble**();**

float ingresarFloat**();**

string ingresarLetra**();**

bool validar**(**string**);**

bool validarEntero**(**string**);**

bool validarLetra**(**string**);**

**};**

int Ingreso**::**ingresarEntero**()** **{**

string numero**;**

bool valido **=** **false;**

**while(!**valido**)** **{**

**try** **{**

getline**(**cin**,**numero**);**

valido **=** validarEntero**(**numero**);**

**if(!**valido**)** **{**

**throw** numero**;**

**}**

**}catch(**string e**)** **{**

cout **<<** "El numero " **<<** e **<<** " no es valido" **<<** endl**;**

**}**

**}**

**return** atoi**(**numero**.**c\_str**());**

**}**

double Ingreso**::**ingresarDouble**()** **{**

string numero**;**

bool valido **=** **false;**

**while(!**valido**)** **{**

**try** **{**

getline**(**cin**,**numero**);**

valido **=** validar**(**numero**);**

**if(!**valido**)** **{**

**throw** numero**;**

**}**

**}catch(**string e**)** **{**

cout **<<** "El numero " **<<** e **<<** " no es valido" **<<** endl**;**

**}**

**}**

**return** atof**(**numero**.**c\_str**());**

**}**

float Ingreso**::**ingresarFloat**()** **{**

string numero**;**

bool valido **=** **false;**

**while(!**valido**)** **{**

**try** **{**

getline**(**cin**,**numero**);**

valido **=** validar**(**numero**);**

**if(!**valido**)** **{**

**throw** numero**;**

**}**

**}catch(**string e**)** **{**

cout **<<** "El numero " **<<** e **<<** " no es valido" **<<** endl**;**

**}**

**}**

**return** atof**(**numero**.**c\_str**());**

**}**

string Ingreso**::**ingresarLetra**()** **{**

string palabra**;**

bool valido **=** **false;**

**while(!**valido**)** **{**

**try** **{**

getline**(**cin**,**palabra**);**

valido **=** validarLetra**(**palabra**);**

**if(!**valido**)** **{**

**throw** palabra**;**

**}**

**}catch(**string e**)** **{**

cout **<<** "La palabra " **<<** e **<<** " no es valida" **<<** endl**;**

**}**

**}**

**return** palabra**;**

**}**

bool Ingreso**::**validar**(**string numero**)** **{**

int inicio **=** 0**;**

**if(**numero**.**length**()** **==** 0**)** **{**

**return** 0**;**

**}**

**if(**numero**[**0**]** **==** '+' **||** numero**[**0**]** **==** '-'**)** **{**

inicio **=** 1**;**

**if(**numero**.**length**()** **==** 1**)** **{**

**return** 0**;**

**}**

**}**

**for(**int i**=**inicio**;** i**<**numero**.**length**();** i**++)** **{**

**if(!**isdigit**(**numero**[**i**])** **&&** numero**[**i**]** **!=** '.'**)** **{**

**return** 0**;**

**}**

**}**

**return** 1**;**

**}**

bool Ingreso**::**validarEntero**(**string numero**)** **{**

int inicio **=** 0**;**

**if(**numero**.**length**()** **==** 0**)** **{**

**return** 0**;**

**}**

**if(**numero**[**0**]** **==** '+' **||** numero**[**0**]** **==** '-'**)** **{**

inicio **=** 1**;**

**if(**numero**.**length**()** **==** 1**)** **{**

**return** 0**;**

**}**

**}**

**for(**int i**=**inicio**;** i**<**numero**.**length**();** i**++)** **{**

**if(!**isdigit**(**numero**[**i**]))** **{**

**return** 0**;**

**}**

**}**

**return** 1**;**

**}**

bool Ingreso**::**validarLetra**(**string palabra**)** **{**

char c**;**

**for(**int i**=**0**;** i**<**palabra**.**size**();** i**++)** **{**

c**=**palabra**[**i**];**

**if(**isalpha**(**c**)** **==** 0**)** **{**

**if(**isspace**(**c**)** **==** 0**)** **{**

**return** 0**;**

**}**

**}**

**}**

**return** 1**;**

**}**

* La librería “generar” realiza gráficamente el codigo QR con el tamaño que nosotros ingresemos

#include <iostream>

#include <time.h>

#include <stdlib.h>

**using** **namespace** std**;**

class QR **{**

private**:**

char **\*\***matriz**;**

int caracter**;**

public**:**

char**\*\*** inicializar**(**int**);**

char**\*\*** encerar**(**int**,**char**\*\*);**

void generarQR**(**int**,**char**\*\*);**

**};**

char**\*\*** QR**::**inicializar**(**int tamanio**)** **{**

matriz **=** **(**char**\*\*)**malloc**(sizeof(**char **\*)\***tamanio**);**

**for(**int i**=**0**;** i**<**tamanio**;** i**++)** **{**

**\*(**matriz**+**i**)** **=** **(**char**\*)**malloc**(sizeof(**char**\*)\***tamanio**);**

**}**

**return** matriz**;**

**}**

char**\*\*** QR**::**encerar**(**int tamanio**,**char **\*\***matriz**)** **{**

**for(**int i**=**0**;** i**<**tamanio**;** i**++)** **{**

**for(**int j**=**0**;** j**<**tamanio**;** j**++)** **{**

**\*(\*(**matriz**+**i**)+**j**)** **=** 0**;**

**}**

**}**

**return** matriz**;**

**}**

void QR**::**generarQR**(**int tamanio**,** char **\*\***matriz**)** **{**

matriz **=** inicializar**(**tamanio**);**

matriz **=** encerar**(**tamanio**,**matriz**);**

cout **<<** ' '**;**

**for(**int i**=**0**;** i**<**tamanio**\***2**;** i**++)** **{**

cout **<<** '-'**;**

**}**

cout **<<** endl**;**

srand**(**time**(NULL));**

**for(**int i**=**0**;** i**<**tamanio**;** i**++)** **{**

cout **<<** '|'**;**

**for(**int j**=**0**;** j**<**tamanio **;** j**++)** **{**

caracter **=** rand**()%(**2**);**

**\*(\*(**matriz**+**i**)+**j**)** **=** caracter**;**

**if(\*(\*(**matriz**+**i**)+**j**)==**0**){**

cout **<<** ' '**;**

**}else** **{**

cout **<<** '\*'**;**

**}**

cout **<<** ' '**;**

**}**

cout **<<** '|' **<<** endl**;**

**}**

cout **<<** ' '**;**

**for(**int i**=**0**;** i**<**tamanio**\***2**;** i**++)** **{**

cout **<<** '-'**;**

**}**

cout **<<** endl**;**

**}**

* Este será nuestro aplicativo donde se ejecutará nuestro programa.

#include "generador.h"

#include <iostream>

#include"ingreso.h"

**using** **namespace** std**;**

int main**()** **{**

Ingreso leer**;**

QR obtener**;**

int tamanio**;**

char **\*\***matriz**;**

cout **<<** "Ingrese el tamanioo que desea el QR" **<<** endl**;**

tamanio **=** leer**.**ingresarEntero**();**

obtener**.**generarQR**(**tamanio**,**matriz**);**

**return** 0**;**

**}**

**Ejecución de la aplicación**

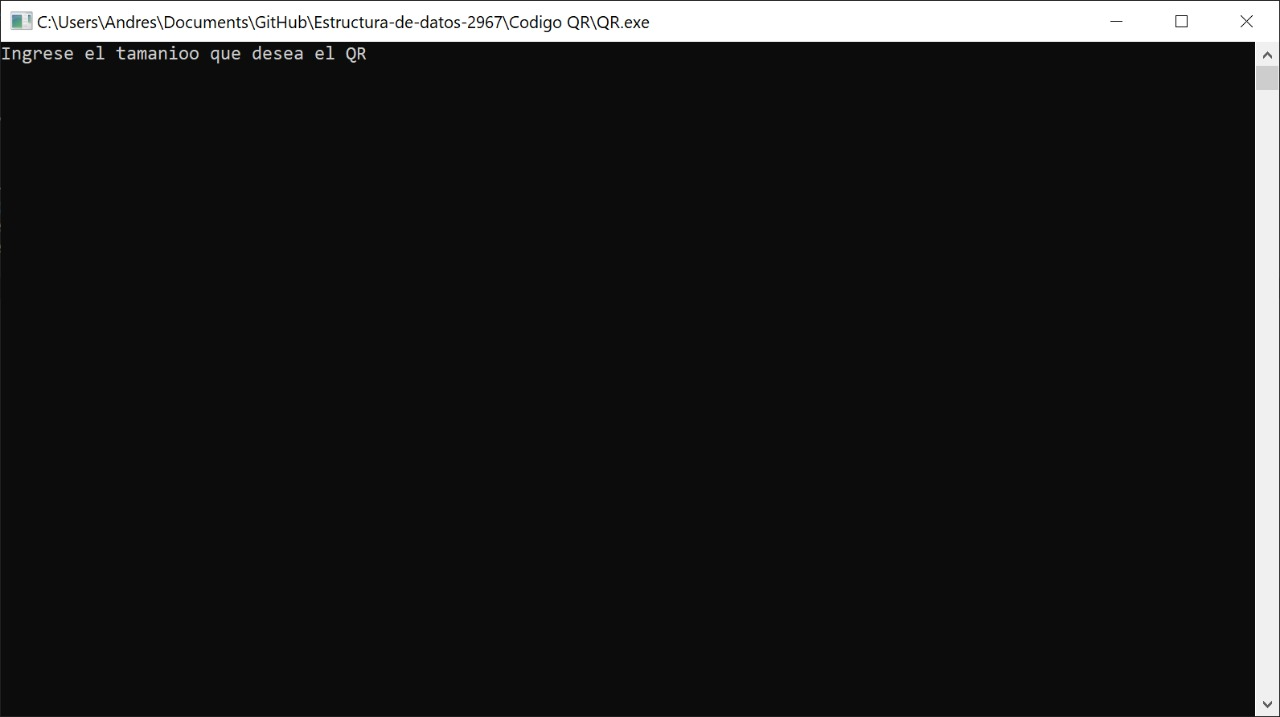
****

Figura 1. Ingreso del tamaño del QR

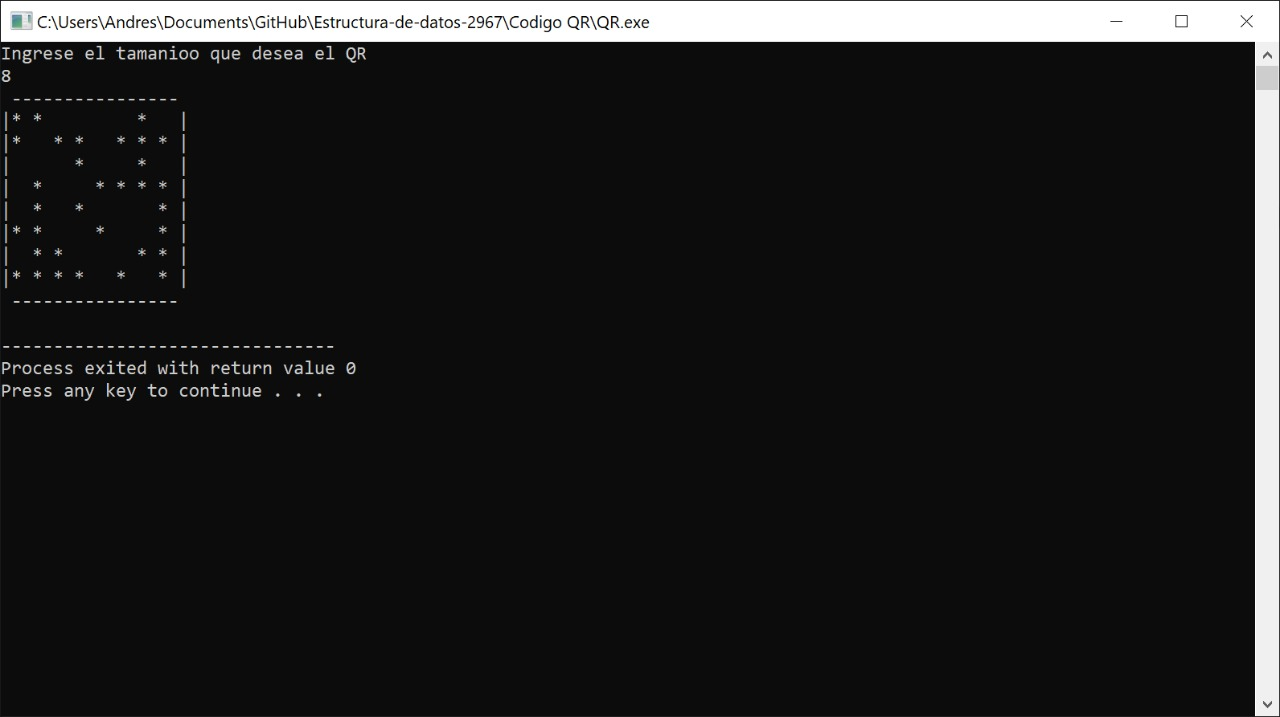


Figura 2. Representación del código QR

1. **Factorial recursivo**

**Descripción**

La operación de factorial aparece en muchas áreas de las matemáticas, particularmente en [combinatoria](https://es.wikipedia.org/wiki/Combinatoria) y [análisis matemático](https://es.wikipedia.org/wiki/An%C3%A1lisis_matem%C3%A1tico). De manera fundamental el factorial de *n* representa el número de formas distintas de ordenar *n* objetos distintos (elementos sin repetición).

El factorial del número entero positivo *n*, denotado n!, se define como el producto de todos los números enteros positivos menores o iguales que n.

n ! = 1 × 2 × 3 × 4 × . . . × ( n − 1 ) × n

**Objetivo de la Aplicación**

* Realizar una función para nuestro aplicativo que nos dé como resultado el factorial de un número usando la definición de recursividad.

**Código de la aplicación**

* A continuación se mostrar la librería “Dato.h” donde se encontrará nuestra función factorial

class Dato

**{**

private**:**

int valor**;**

public**:**

Dato**(**int**);**

Dato**();**

int getValor**();**

void setValor**(**int**);**

int factorial**(**int**);**

**};**

Dato**::**Dato**(**int valor**)**

**{**

**this->**valor**=**valor**;**

**}**

Dato**::**Dato**()**

**{**

**this->**valor**=**0**;**

**}**

int Dato**::**getValor**()**

**{**

**return** valor**;**

**}**

void Dato**::**setValor**(**int valor**)**

**{**

**this->**valor**=**valor**;**

**}**

int Dato**::**factorial**(**int valor**)**

**{**

**if(**valor**==**1**)**

**{**

**return** 1**;**

**}**

**else{**

**return** valor**\***factorial**(**valor**-**1**);**

**}**

**}**

* Este será nuestro main para ejecutar el aplicativo

#include<iostream>

#include "Ingreso.h"

#include<stdlib.h>

#include "Dato.h"

**using** **namespace** std**;**

int main**()**

**{**

Ingreso leer**;**

char valor**;**

Dato dato**=**Dato**();**

dato**.**setValor**(**leer**.**ingresarInt**(&**valor**));**

dato**.**setValor**(**dato**.**factorial**(**dato**.**getValor**()));**

cout**<<**dato**.**getValor**()<<**endl**;**

system**(**"pause"**);**

**return** 0**;**

**}**

**Ejecución de la aplicación**

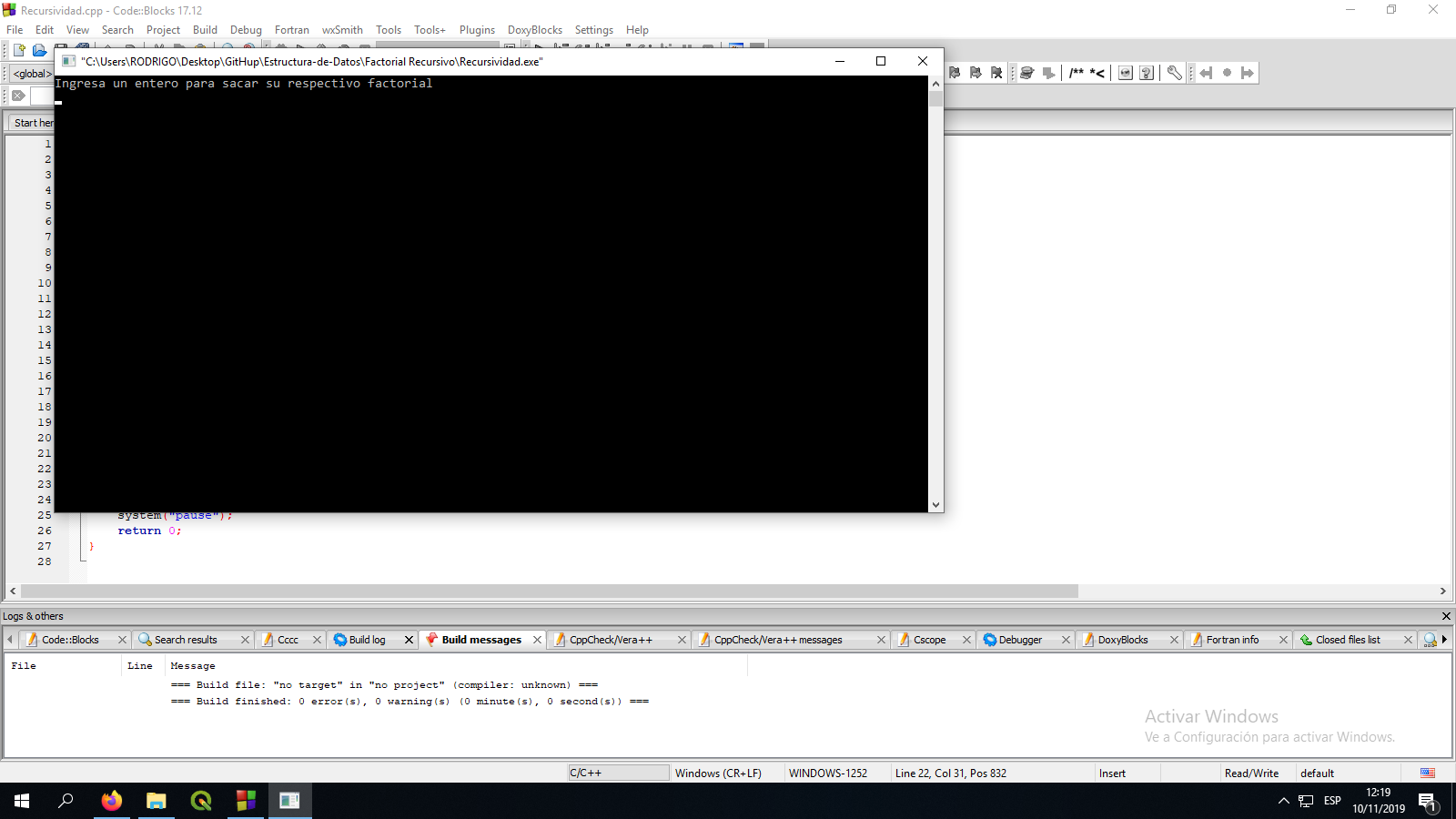


Figura 1. Ejecución de la aplicación

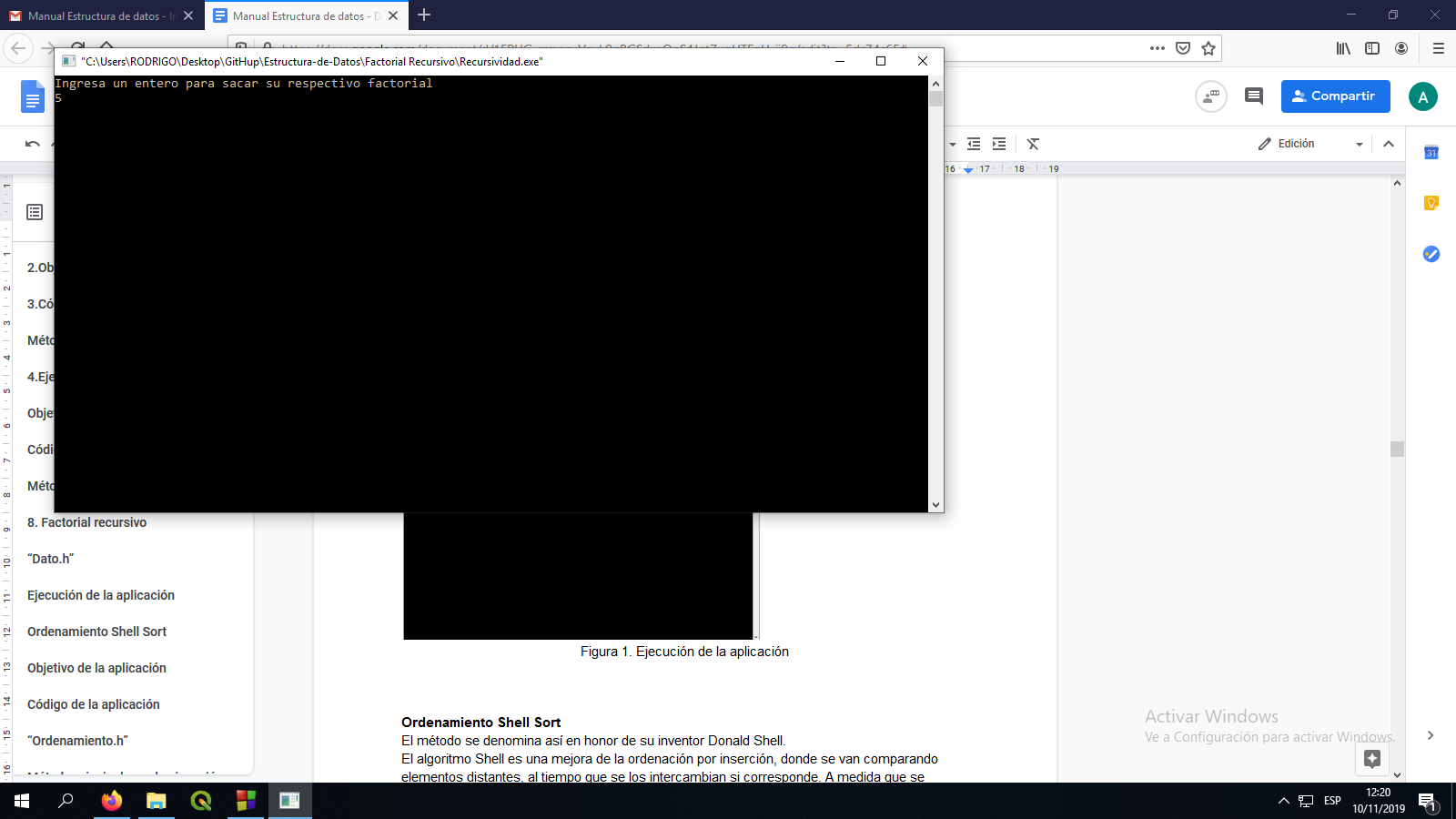


Figura 2. Inserción de Datos

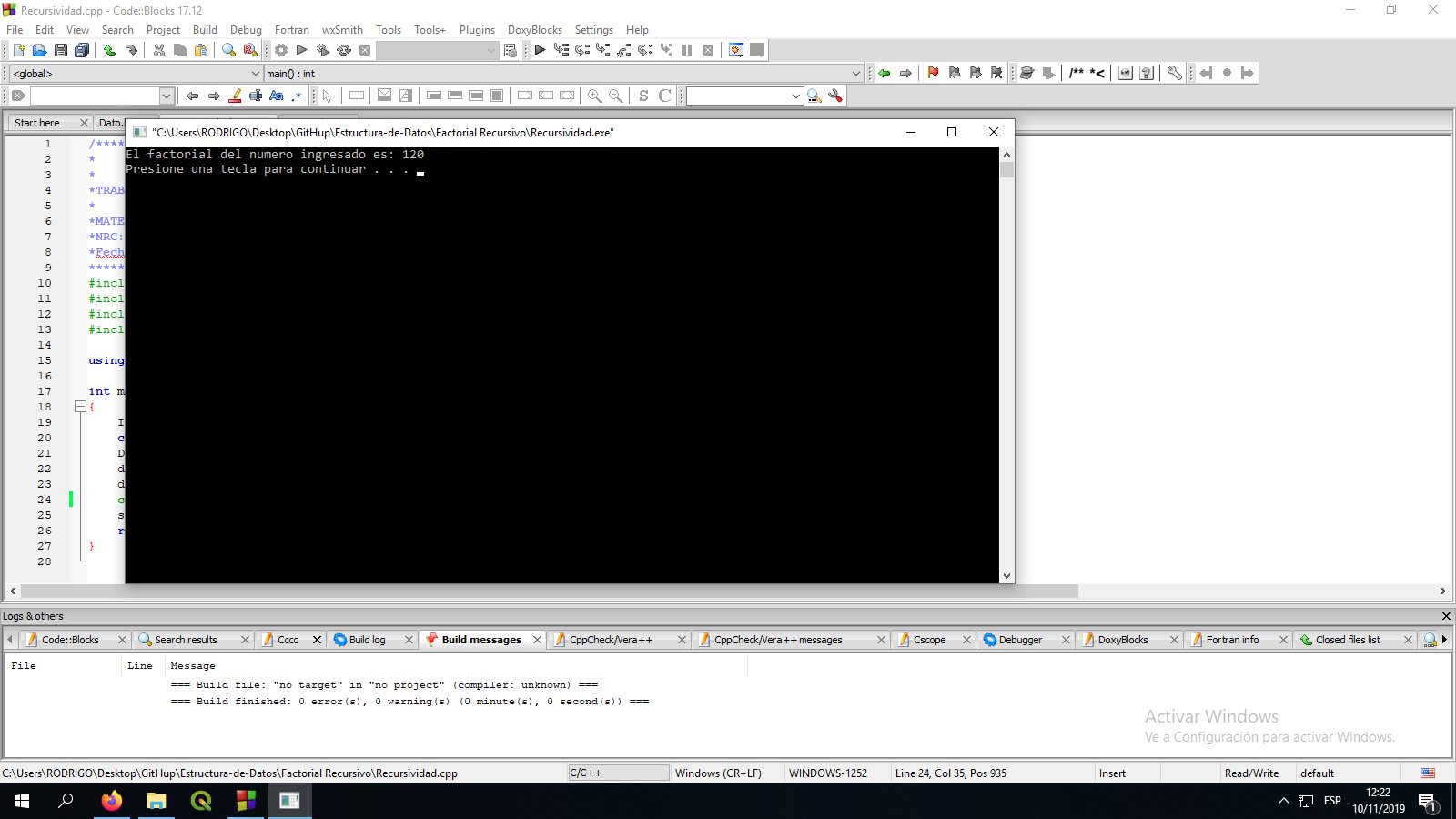


Figura 3. Factorial Recursivo

**8. Ordenamiento por Quicksort**

**Descripción**

Es un algoritmo creado por el científico británico en computación Tony Hoare y basado en la técnica de divide y vencerás. Esta es la técnica quizás la más eficiente y en ella que en la mayoría de los casos da mejores resultados

**Objetivo de la aplicación**

El objetivo de estudiar los algoritmos de ordenamiento es que permite ejemplificar la importancia del estudio de la eficiencia de los algoritmos tanto en rapidez como en implementación lleva este método.

**Código de la aplicación**

* En la librería “Quicksort.h” se encuentran todas las funciones necesarias para que el proceso de ordenamiento se realice de una manera correcta.

include **<**iostream**>**

#include <stdlib.h>

#include "Ingreso.h"

**using** **namespace** std**;**

void leeCadena**(**int cant**,** int**\*** n**)** **{**

Ingreso ingreso**;**

**for** **(**int i **=** 0**;** i **<** cant**;** i**++)** **{**

**\*(**n **+** i**)** **=** atoi**(**ingreso**.**ingresar**(**"Ingrese valor entero: "**).**c\_str**());**

**}**

**}**

void muestraCadena**(**int cant**,** int**\*** n**)** **{**

cout **<<** "Cadena ordenada por Quicksort" **<<** endl**;**

**for** **(**int i **=** 0**;** i **<** cant**;** i**++)** **{**

printf**(**"%d "**,** **\*(**n **+** i**));**

**}**

**}**

void quicksort**(**int**\*** A**,** int izq**,** int der**)** **{**

int aux**;**

int i **=** izq**;**

int j **=** der**;**

int x **=** **\*(**A **+** **((**izq **+** der**)** **/** 2**));**

**do** **{**

**while** **((\*(**A **+** i**)** **<** x**)** **&&** **(**j **<=** der**))** **{**

i**++;**

**}**

**while** **((**x **<** **\*(**A **+** j**))** **&&** **(**j **>** izq**))** **{**

j**--;**

**}**

**if** **(**i **<=** j**)** **{**

aux **=** **\*(**A **+** i**);**

**\*(**A **+** i**)** **=** **\*(**A **+** j**);**

**\*(**A **+** j**)** **=** aux**;**

i**++;**

j**--;**

**}**

**}** **while** **(**i **<=** j**);**

**if** **(**izq **<** j**)**

quicksort**(**A**,** izq**,** j**);**

**if** **(**i **<** der**)**

quicksort**(**A**,** i**,** der**);**

**}**

* La librería “Ingreso.h” es la que permite que el usuario digite los datos que van a ser procesados para el cálculo de las funciones trigonométricas.

#ifndef INGRESO\_H

#define INGRESO\_H

#include <iostream>

#include <string>

**using** **namespace** std**;**

class Ingreso **{**

public**:**

string ingresar**(**string**);**

bool validarTipoFloat**(**string**);**

bool validarTipoInt**(**string**);**

bool validarTipoString**(**string**);**

**};**

string Ingreso**::**ingresar**(**string msg**)** **{**

string dato\_a\_validar**;**

bool esValido **=** **false;**

**while** **(!**esValido**)** **{**

**try** **{**

cout **<<** msg**;**

getline**(**cin**,** dato\_a\_validar**);**

**if** **(**msg**.**find**(**"flotante"**)** **!=** std**::**string**::**npos**)** **{**

esValido **=** validarTipoFloat**(**dato\_a\_validar**);**

**}**

**else** **if** **(**msg**.**find**(**"entero"**)** **!=** std**::**string**::**npos**)** **{**

esValido **=** validarTipoInt**(**dato\_a\_validar**);**

**}**

**else** **if** **(**msg**.**find**(**"cadena"**)** **!=** std**::**string**::**npos**)** **{**

esValido **=** validarTipoString**(**dato\_a\_validar**);**

**}**

**if** **(!**esValido**)** **{**

**throw** dato\_a\_validar**;**

**}**

**}**

**catch** **(**string e**)** **{**

cout **<<** "El dato (" **<<** e **<<** ") no es valido" **<<** endl**;**

**}**

**}**

**return** dato\_a\_validar**;**

**}**

bool Ingreso**::**validarTipoFloat**(**string numero**)** **{**

int inicio **=** 0**;**

**if** **(**numero**.**length**()** **==** 0**)** **{**

**return** 0**;**

**}**

**if** **(**numero**[**0**]** **==** '+' **||** numero**[**0**]** **==** '-'**)** **{**

inicio **=** 1**;**

**if** **(**numero**.**length**()** **==** 1**)** **{**

**return** 0**;**

**}**

**}**

**for** **(**int i **=** inicio**;** i **<** numero**.**length**();** i**++)** **{**

**if** **(!**isdigit**(**numero**[**i**])** **&&** numero**[**i**]** **!=** '.'**)** **{**

**return** 0**;**

**}**

**}**

**return** 1**;**

**}**

bool Ingreso**::**validarTipoInt**(**string numero**)** **{**

int inicio **=** 0**;**

**if** **(**numero**.**length**()** **==** 0**)** **{**

**return** 0**;**

**}**

**if** **(**numero**[**0**]** **==** '+' **||** numero**[**0**]** **==** '-'**)** **{**

inicio **=** 1**;**

**if** **(**numero**.**length**()** **==** 1**)** **{**

**return** 0**;**

**}**

**}**

**for** **(**int i **=** inicio**;** i **<** numero**.**length**();** i**++)** **{**

**if** **(!**isdigit**(**numero**[**i**]))** **{**

**return** 0**;**

**}**

**}**

**return** 1**;**

**}**

bool Ingreso**::**validarTipoString**(**string numero**)** **{**

**if** **(**numero**.**length**()** **==** 0**)** **{**

**return** 0**;**

**}**

**for** **(**int i **=** 0**;** i **<** numero**.**length**();** i**++)** **{**

**if** **(**isdigit**(**numero**[**i**]))** **{**

**return** 0**;**

**}**

**}**

**return** 1**;**

**}**

* Este será nuestro main para ejecutar nuestro aplicativo.

#include <iostream>

#include "Quicksort.h"

#include "Ingreso.h"

**using** **namespace** std**;**

int main**()** **{**

Ingreso ingreso**;**

int n **=** atoi**(**ingreso**.**ingresar**(**"Indique valor entero de numeros a ingresar: "**).**c\_str**());**

int**\*** A **=** **(**int**\*)** malloc**(**n **\*** **sizeof** **(**int**));**

leeCadena**(**n**,** A**);**

quicksort**(**A**,** 0**,** n **-** 1**);**

muestraCadena**(**n**,** A**);**

**return** 0**;**

**}**

**Ejecución del Aplicativo**

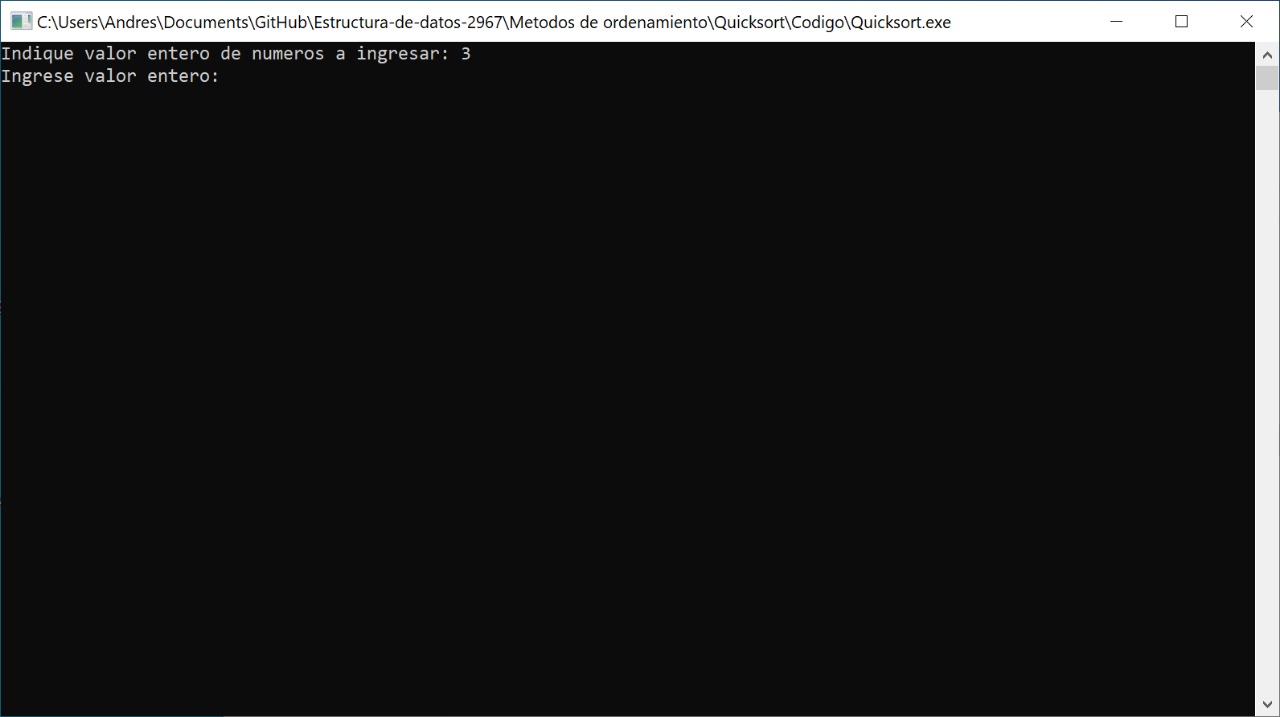
****

Figura 1: Se ingresa los valores a ordenar

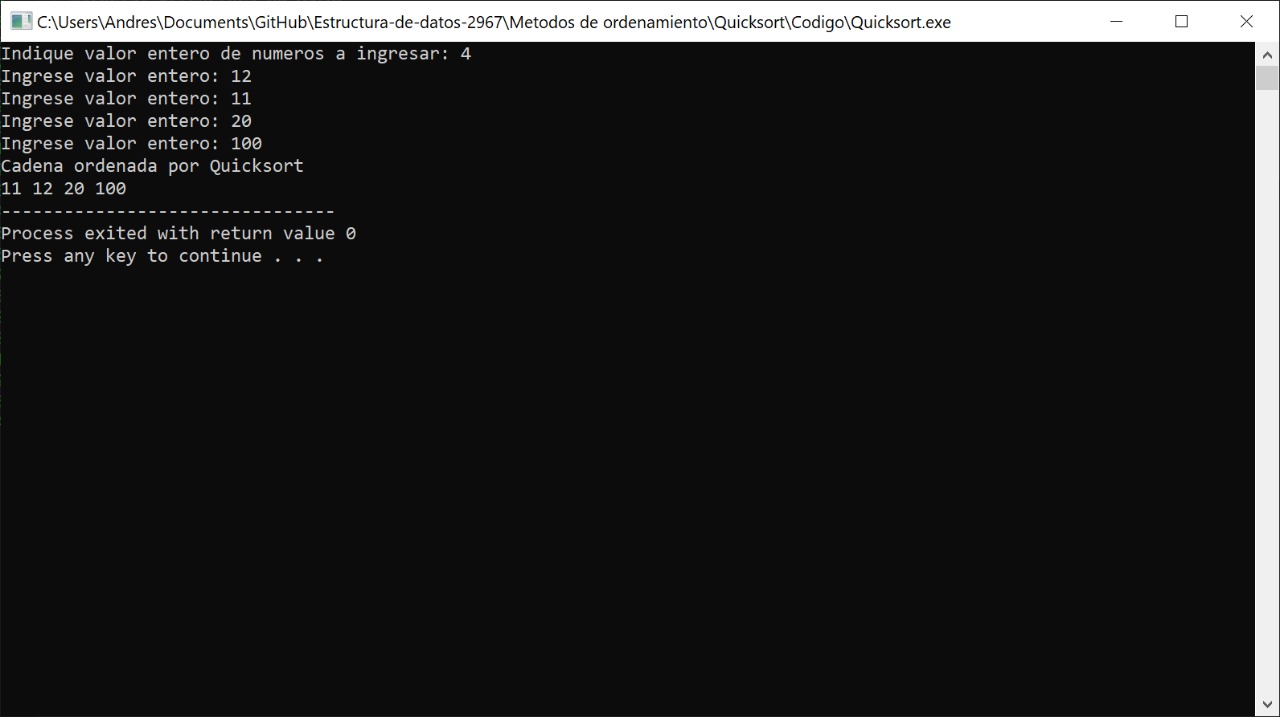


Figura 2: Indica la cadena de caracteres ordenados

1. **Ordenamiento ShellSort**

**Descripción**

El método se denomina así en honor de su inventor Donald Shell.

El algoritmo Shell es una mejora de la ordenación por inserción, donde se van comparando elementos distantes, al tiempo que se los intercambian si corresponde. A medida que se aumentan los pasos, el tamaño de los saltos disminuye; por esto mismo, es útil tanto como si los datos desordenados se encuentran cercanos, o lejanos.

Es bastante adecuado para ordenar listas de tamaño moderado, debido a que su velocidad es aceptable y su codificación es bastante sencilla. Su velocidad depende de la secuencia de valores con los cuales trabaja, ordenándolos.

**Objetivo de la aplicación**

Ordenar los elementos ingresados por el usuario mediante el uso del algoritmo shellsort.

**Código de la aplicación**

* A continuación mostraremos la librería “Ordenamiento.h” la cual contiene los prototipos de nuestro algoritmo de ordenamiento Shell sort

#include <iostream>

#include <stdlib.h>

#include "Ingreso.h"

**using** **namespace** std**;**

class Ordenamiento **{**

public**:**

void ingresarDatos**(**int dim**,** int**\*** arreglo**);**

void imprimir**(**int**\*** arreglo**,** int dim**);**

void ordenarShell**(**int**\*** arreglo**,** int dim**);**

int**\*** inicializarVector**(**int dim**);**

Ordenamiento**();**

**~**Ordenamiento**();**

protected**:**

private**:**

**};**

* En la clase “Ordenamiento.cpp” se crear los metodos declarados en “Ordenamiento.h”.

#include "Ordenamiento.h"

void Ordenamiento**::**ingresarDatos**(**int dim**,** int**\*** arreglo**)** **{**

Ingreso ingreso**;**

**for** **(**int i **=** 0**;** i **<** dim**;** i**++)** **{**

**\*(**arreglo **+** i**)** **=** atoi**(**ingreso**.**ingresar**(**"Ingrese valor entero: "**).**c\_str**());**

**}**

**}**

void Ordenamiento**::**imprimir**(**int**\*** arreglo**,** int dim**)** **{**

**for** **(**int i **=** 0**;** i **<** dim**;** i**++)**

cout **<<** **\*(**arreglo **+** i**)** **<<** " "**;**

**}**

int**\*** Ordenamiento**::**inicializarVector**(**int dim**)** **{**

int**\*** a**;**

a **=** **(**int**\*)** calloc**(**dim**,** **sizeof** **(**int**));**

**return** a**;**

**}**

void Ordenamiento**::**ordenarShell**(**int**\*** arreglo**,** int dim**)** **{**

**for** **(**int gap **=** dim **/** 2**;** gap **>** 0**;** gap **/=** 2**)** **{**

**for** **(**int i **=** gap**;** i **<** dim**;** i **+=** 1**)** **{**

int temp **=** **\*(**arreglo **+** i**);**

int j**;**

**for** **(**j **=** i**;** j **>=** gap **&&** **\*(**arreglo **+** **(**j **-** gap**))** **>** temp**;** j **-=** gap**)**

**\*(**arreglo **+** j**)** **=** **\*(**arreglo **+** **(**j **-** gap**));**

**\*(**arreglo **+** j**)** **=** temp**;**

**}**

**}**

**}**

Ordenamiento**::**Ordenamiento**()** **{**

**}**

Ordenamiento**::~**Ordenamiento**()** **{**

**}**

* Este será nuestro main para ejecutar el aplicativo.

#include <iostream>

#include "Ordenamiento.cpp"

**using** **namespace** std**;**

int main**(**int argc**,** char**\*\*** argv**)** **{**

Ingreso ingreso**;**

Ordenamiento ordenar**;**

int**\*** arreglo**;**

int dimension **=** atoi**(**ingreso**.**ingresar**(**"Indique valor entero de numeros a ingresar: "**).**c\_str**());**

arreglo **=** ordenar**.**inicializarVector**(**dimension**);**

ordenar**.**ingresarDatos**(**dimension**,** arreglo**);**

cout **<<** "\nArreglo antes de ordenar\n"**;**

ordenar**.**imprimir**(**arreglo**,** dimension**);**

ordenar**.**ordenarShell**(**arreglo**,** dimension**);**

cout **<<** "\nArreglo ordenado con el metodo ShellSort\n"**;**

ordenar**.**imprimir**(**arreglo**,** dimension**);**

free**(**arreglo**);**

**return** 0**;**

**}**

**Ejecución de la aplicación**

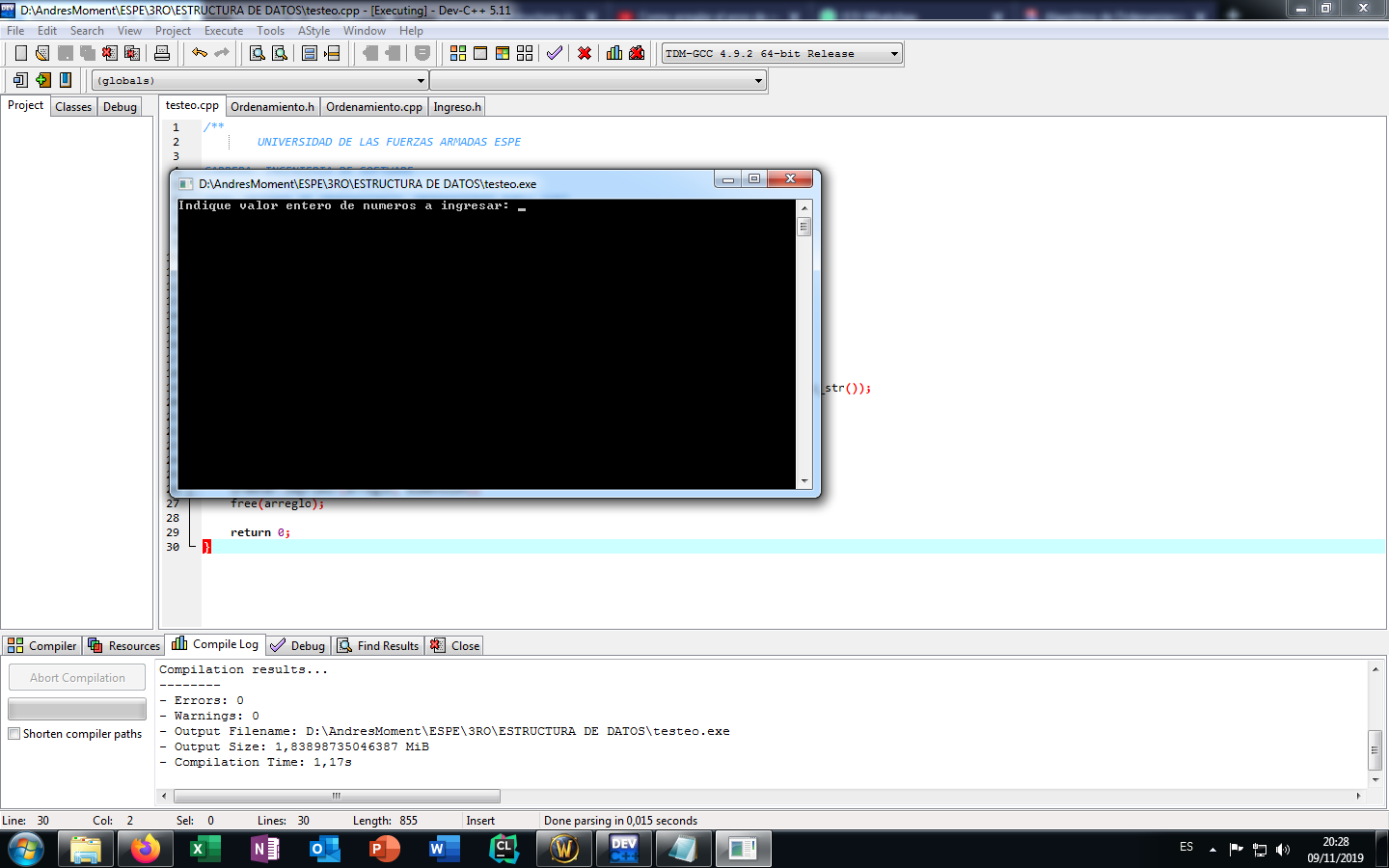


Figura 1. Ejecución de la aplicación

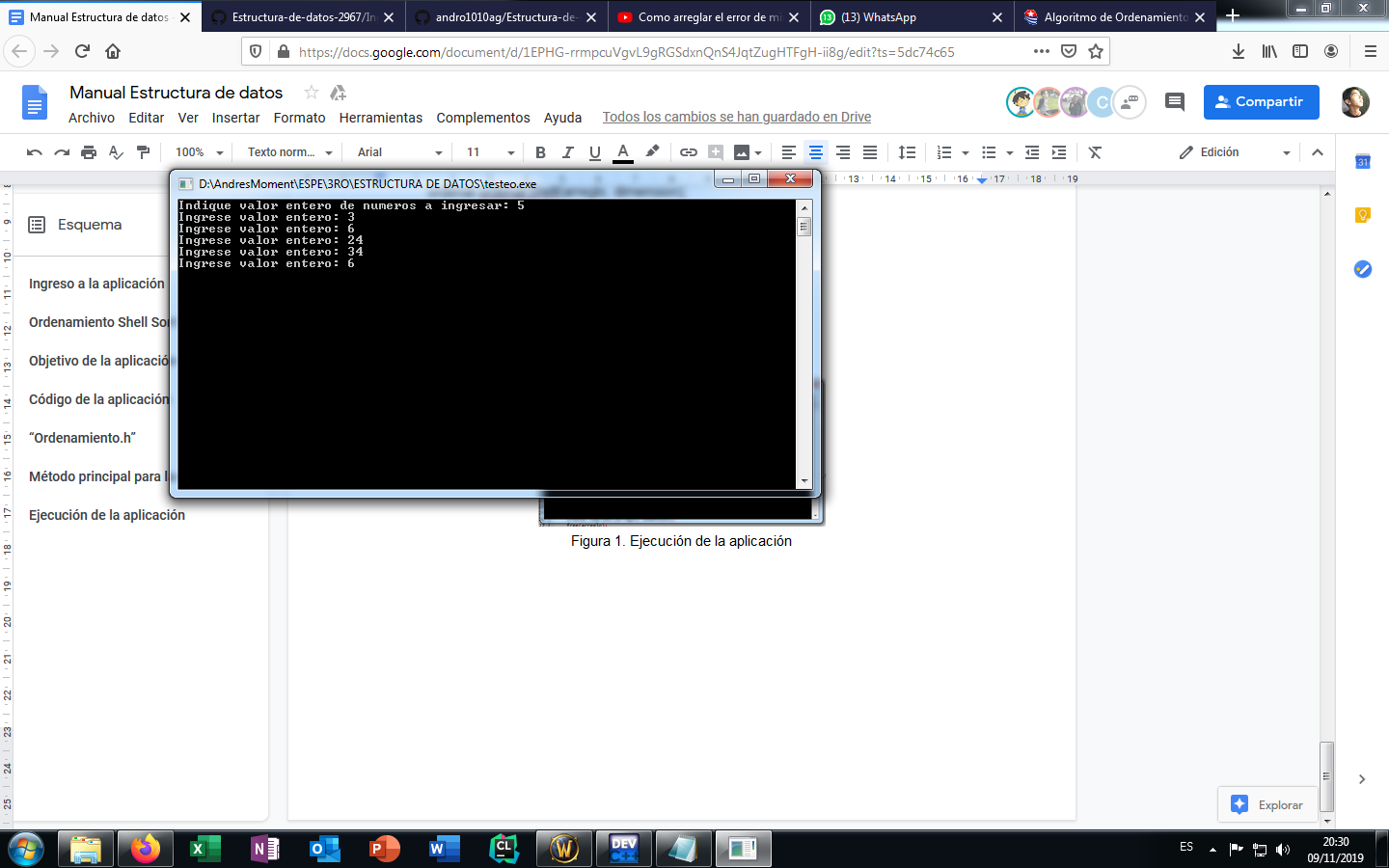


Figura 2. Inserción de Datos

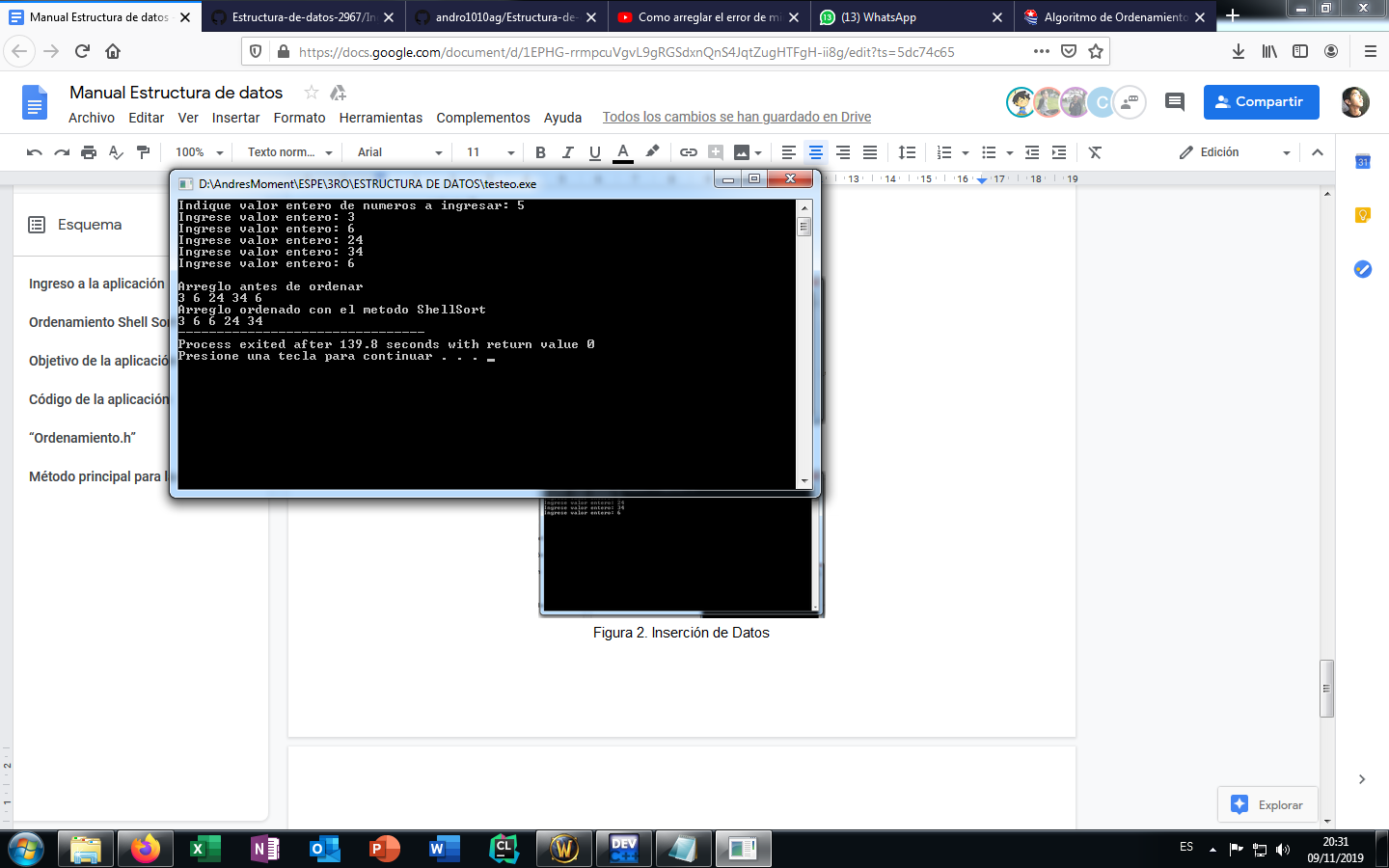


Figura 3. Ordenamiento Shell

1. **Método de Ordenamiento Heapsort**

**Descripción**

Es un método de ordenamiento basado con comparación, usa el Montículo o Heap como estructura de datos. Este método es más lento que otros métodos, pero es más eficaz en escenarios más rigurosos.

**Objetivo de la aplicación**

Ordenar los elementos de un vector utilizando el método heapsort

**Código de la aplicación**

* A continuación mostraremos el código necesario para la aplicación:

#include <iostream>

**using** **namespace** std**;**

class Ingreso **{**

public**:**

string ingresar**(**string msg**);**

**};**

string Ingreso**::**ingresar**(**string msg**)** **{**

string valor**;**

cout **<<** msg **<<** endl**;**

cin **>>** valor**;**

**return** valor**;**

**}**

* Ahora para validar datos tenemos la siguiente librería:

#include <iostream>

**using** **namespace** std**;**

class Validacion

**{**

public**:**

int validar**(**string val**);**

**};**

int Validacion**::**validar**(**string val**)** **{**

int validez **=** 0**;**

int punto **=** 0**;**

int letras **=** 0**;**

**for** **(**int i **=** 0**;** i **<** val**.**size**();** i**++)**

**{**

**if** **(!**isdigit**(**val**[**i**]))** **{**

letras**++;**

**}**

**if** **(**val**[**i**]** **==** '.'**)** **{**

punto**++;**

**}**

**}**

**if** **((**letras **-** punto**)** **>** 0 **||** punto **>** 1**)** **{**

validez **=** 1**;**

**}**

**return** validez**;**

**}**

* Ahora tenemos la clase que tiene el algoritmo de ordenamiento heapsort

#include <iostream>

#include <stdlib.h>

#include <time.h>

**using** **namespace** std**;**

class Ordenar**{**

private**:**

int **\***matriz**;**

public**:**

Ordenar**();**

void setMatriz**(**int **\*);**

int**\*** getMatriz**();**

void inicializarMatriz**(**Ordenar**,**int**);**

void heapify**(**Ordenar**,**int **,** int **);**

void heapSort**(**Ordenar **,** int **);**

void imprimir**(**Ordenar **,** int **);**

void llenar**(**Ordenar **,**int **);**

**};**

Ordenar**::**Ordenar**(){**

**}**

void Ordenar**::**setMatriz**(**int **\***v**){**

matriz **=** v**;**

**}**

int**\*** Ordenar**::**getMatriz**(){**

**return** matriz**;**

**}**

void Ordenar**::**inicializarMatriz**(**Ordenar v**,**int tam**){**

v**.**matriz **=** **(**int**\*)**malloc**(**tam**\*sizeof(**int**));**

v**.**setMatriz**(**v**.**matriz**);**

**}**

void Ordenar**::**heapify**(**Ordenar ord**,** int n**,** int i**){**

// Encuentra más grande entre la raíz, el hijo izquierdo y el derecho

int mayor **=** i**;**

int izquierda **=** 2**\***i **+** 1**;**

int derecha**=** 2**\***i **+** 2**;**

**if** **(**izquierda**<** n **&&** **\*(**ord**.**matriz**+**izquierda**)>** **\*(**ord**.**matriz**+**mayor**))**

mayor **=** izquierda**;**

**if** **((**derecha**<**n**)&&** **(\*(**ord**.**matriz**+**derecha**)>\*(**ord**.**matriz**+**mayor**)))**

mayor **=** derecha**;**

// Intercambiar y continuar la heapificación si la raíz no es más grande

**if** **(**mayor **!=** i**)**

**{**

swap**(\*(**ord**.**matriz**+**i**),** **\*(**ord**.**matriz**+**mayor**));**

heapify**(**ord**,** n**,** mayor**);**

**}**

**}**

// Función principal para hacer la ordenación del montón

void Ordenar**::** heapSort**(**Ordenar ord**,** int n**){**

// Construir el montón máximo

**for** **(**int i **=** n **/** 2 **-** 1**;** i **>=** 0**;** i**--)**

heapify**(**ord**,** n**,** i**);**

// tipo de pila

**for** **(**int i**=**n**-**1**;** i**>=**0**;** i**--)**

**{**

swap**(\*(**ord**.**matriz**+**0**),** **\*(**ord**.**matriz**+**i**));**

// Heapify elemento raíz para obtener el elemento más alto en la raíz de nuevo

heapify**(**ord**,** i**,** 0**);**

**}**

**}**

void Ordenar**::** imprimir **(**Ordenar ord**,** int n**)**

**{**

**for** **(**int i**=**0**;** i**<**n**;** **++**i**){**

cout **<<** **\*(**ord**.**matriz**+**i**)** **<<** " "**;**

cout **<<** " "**;**

**}**

**}**

void Ordenar**::**llenar**(**Ordenar ord**,**int tam**){**

srand**(**time**(NULL));**

**for(**int i**=**0**;**i**<**tam**;**i**++){**

**\*(**ord**.**matriz**+**i**)=**1**+**rand**()%(**100**-**1**);**

cout**<<\*(**ord**.**matriz**+**i**)<<**" "**;**

**}**

cout**<<**endl**;**

**}**

* Finalmente tenemos el método principal (Main) en el que usaremos las librerías ya descritas anteriormente.

#include <iostream>

#include <sstream>

#include "Ingreso.h"

#include "Validacion.h"

#include "heapsort.h"

**using** **namespace** std**;**

int main**(**int argc**,** char**\*\*** argv**)** **{**

Ingreso ingreso**;**

Validacion validacion**;**

string numero**;**

int val**;**

int tam**;**

int **\***matr**;**

Ordenar ordenar**;**

**do{**

numero**=**ingreso**.**ingresar**(**"Ingrese el tamano de la matriz: "**);**

val**=**validacion**.**validar**(**numero**);**

**}while(**val**==**1**);**

istringstream **(** numero **)** **>>** tam**;**

matr**=** **(**int**\*)**malloc**(**tam**\*sizeof(**int**));**

ordenar**.**setMatriz**(**matr**);**

cout**<<**"Arreglo Desordenado"**<<**endl**;**

cout**<<**endl**;**

ordenar**.**llenar**(**ordenar**,**tam**);**

cout**<<**endl**;**

ordenar**.**heapSort**(**ordenar**,** tam**);**

cout **<<** "Arreglo Ordenado \n"**;**

cout**<<**endl**;**

ordenar**.**imprimir**(**ordenar**,** tam**);**

cout**<<**endl**;**

system**(**"PAUSE"**);**

**return** 0**;**

**}**

**Ejecución de la aplicación**

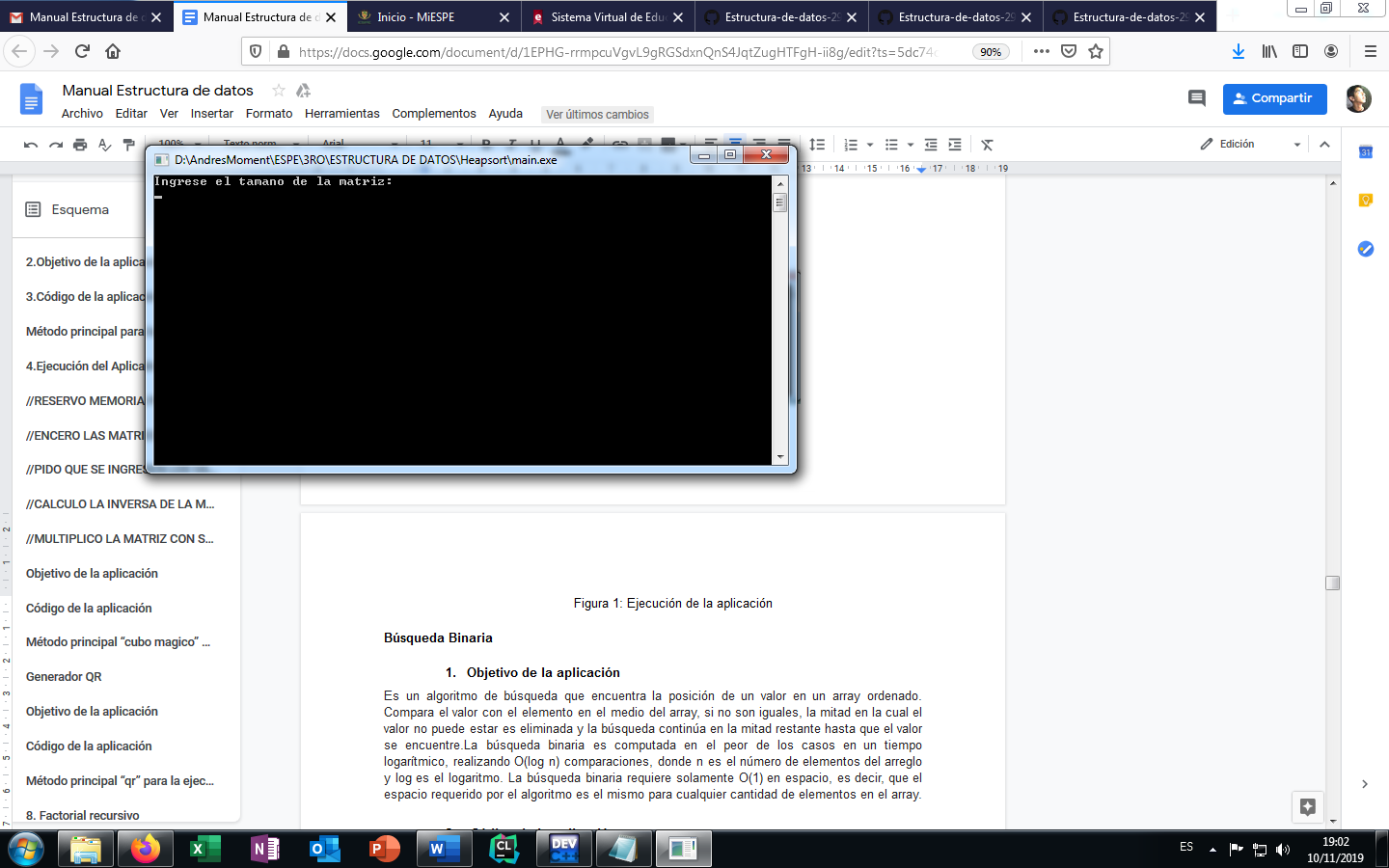


Figura 1: Ejecución de la aplicación

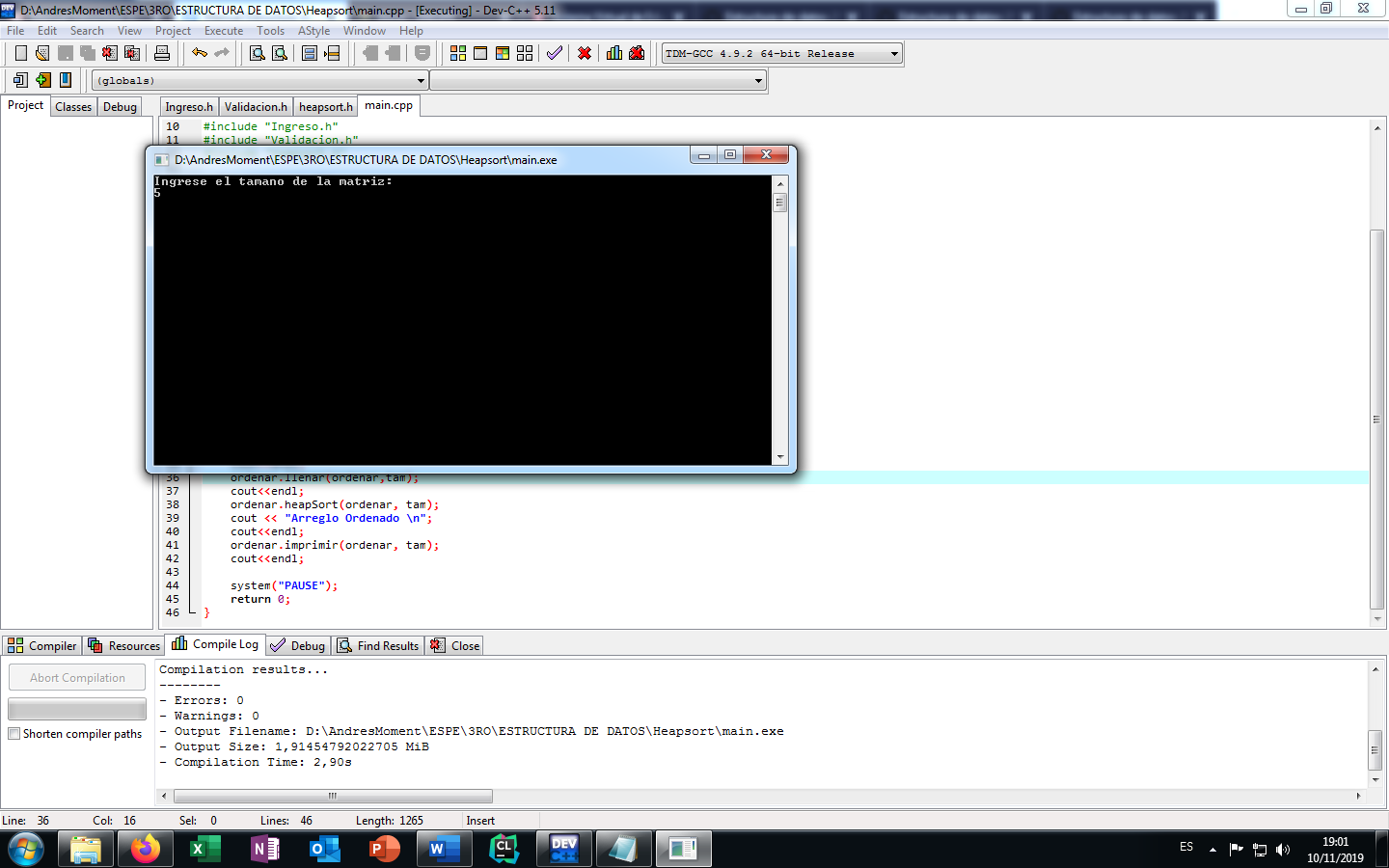
****

Figura 2: Ingreso de Datos

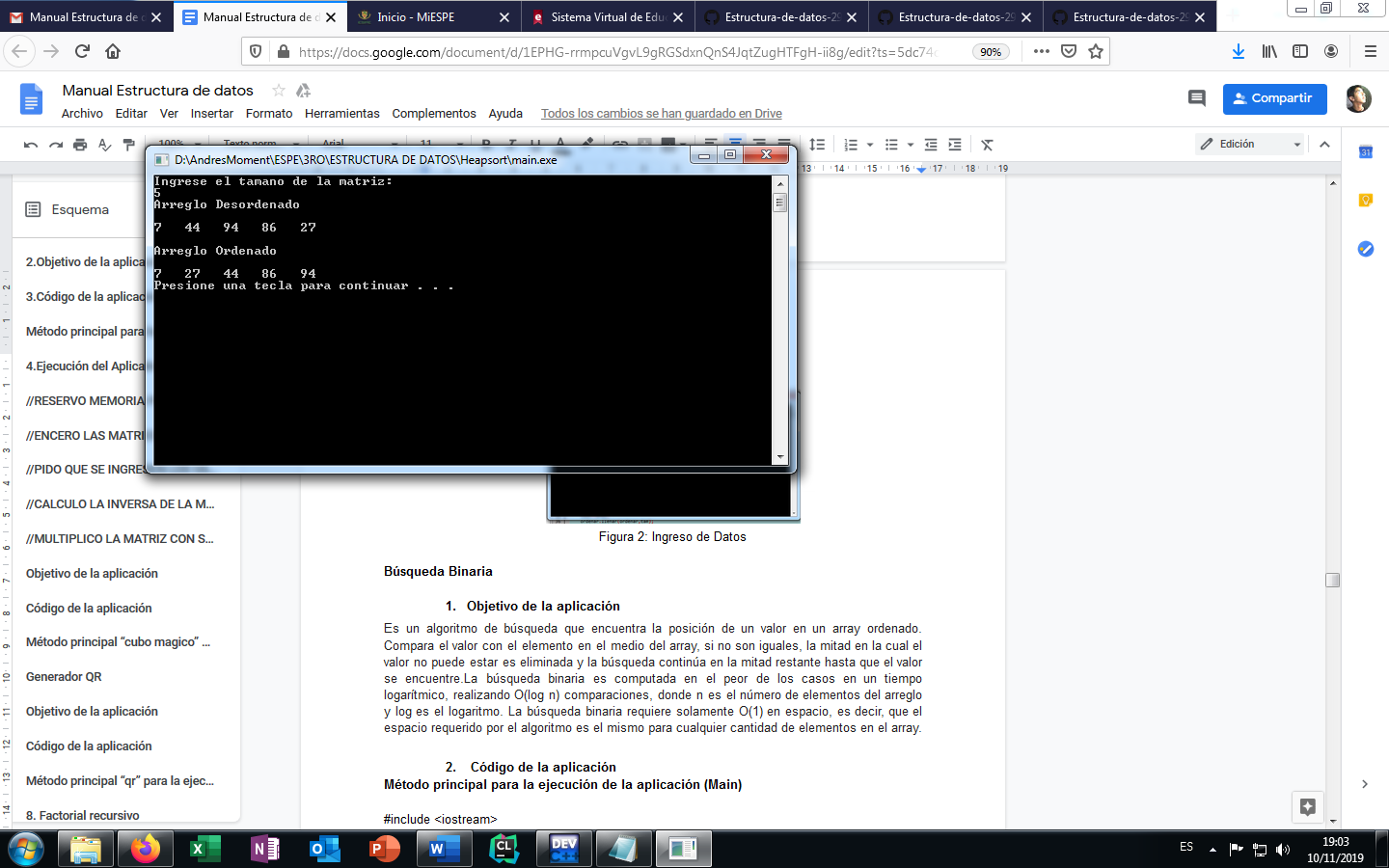


Figura 3: Ordenamiento Heapsort

1. **Búsqueda Binaria**

**Descripción**

Es un algoritmo de búsqueda que encuentra la posición de un valor en un array ordenado.Compara el valor con el elemento en el medio del array, si no son iguales, la mitad en la cual el valor no puede estar es eliminada y la búsqueda continúa en la mitad restante hasta que el valor se encuentre. La búsqueda binaria es computada en el peor de los casos en un tiempo logarítmico, realizando O(log n) comparaciones, donde n es el número de elementos del arreglo y log es el logaritmo. La búsqueda binaria requiere solamente O(1) en espacio, es decir, que el espacio requerido por el algoritmo es el mismo para cualquier cantidad de elementos en el array.​

**Código de la aplicación**

* Aquí se encuentra el main de la aplicación que nos permite ejecutarla

#include <iostream>

#include "Busqueda.h"

**using** **namespace** std**;**

int main**(**int argc**,** char**\*\*** argv**)** **{**

Ingreso leer**;**

Busqueda b**;**

int dim**,** numero**,** pos**;**

int**\*** vector**;**

int**\*** vectorOrdenado**;**

vector **=** b**.**inicializarVector**(**dim**);**

vectorOrdenado **=** b**.**inicializarVector**(**dim**);**

cout **<<** "Ingrese la dimension deseada del arreglo: " **<<** endl**;**

dim **=** leer**.**ingresarEntero**();**

b**.**ingresoDatos**(**dim**,** vector**);**

vectorOrdenado **=** b**.**ordenamientoBurbuja**(**dim**,** vector**);**

cout **<<** "Ingrese el elemento a buscar:" **<<** endl**;**

numero **=** leer**.**ingresarEntero**();**

pos **=** b**.**busquedaBinaria**(**numero**,** 0**,** dim **-** 1**,** vectorOrdenado**);**

cout **<<** "El elemento: " **<<** "[" **<<** numero **<<** "]" **<<** " se encuentra en la posicion: " **<<** "[" **<<** pos **<<** "]" **<<** " del arreglo" **<<** endl**;**

free**(**vector**);**

free**(**vectorOrdenado**);**

**return** 0**;**

**}**

* Se implementan los métodos generados en la clase “Busqueda.h”.

#include "Busqueda.h"

int**\*** Busqueda**::**ordenamientoBurbuja**(**int dim**,** int**\*** a**)** **{**

int aux**;**

**for** **(**int i **=** 0**;** i **<** dim **-** 1**;** i**++)** **{**

**for** **(**int j **=** 0**;** j **<** dim **-** 1**;** j**++)** **{**

**if** **(\*(**a **+** j**)>\*(**a **+** j **+** 1**))** **{**

aux **=** **\*(**a **+** j**);**

**\*(**a **+** j**)** **=** **\*(**a **+** j **+** 1**);**

**\*(**a **+** j **+** 1**)** **=** aux**;**

**}**

**}**

**}**

cout **<<** "El arreglo ordenado por burbuja es: " **<<** endl**;**

**for** **(**int i **=** 0**;** i **<** dim**;** i**++)** **{**

cout **<<** "[" **<<** **\*(**a **+** i**)** **<<** "]" **<<** " "**;**

**}**

cout **<<** endl**;**

**return** a**;**

**}**

int**\*** Busqueda**::**inicializarVector**(**int dim**)** **{**

int**\*** a**;**

a **=** **(**int**\*)** calloc**(**dim**,** **sizeof** **(**int**));**

**return** a**;**

**}**

int**\*** Busqueda**::**ingresoDatos**(**int dim**,** int**\*** a**)** **{**

Ingreso leer**;**

int aux**;**

**for** **(**int i **=** 0**;** i **<** dim**;** i**++)** **{**

cout **<<** "Ingrese el elemento de la posicion: " **<<** "[" **<<** i **<<** "]" **<<** endl**;**

aux **=** leer**.**ingresarEntero**();**

**\*(**a **+** i**)** **=** aux**;**

**}**

**}**

int Busqueda**::**busquedaBinaria**(**int x**,** int inicio**,** int final**,** int **\***array**)** **{**

int q**;**

q **=** **(**inicio **+** final**)** **/** 2**;**

**if** **(**x **==** **\*(**array **+** q**))** **{**

**return** q**;**

**}** **else** **if** **(**x**>\*(**array **+** q**))** **{**

busquedaBinaria**(**x**,** q **+** 1**,** final**,** array**);**

**}**

**else** **if** **(**x**<\*(**array **+** q**))** **{**

busquedaBinaria**(**x**,** inicio**,** q **-** 1**,** array**);**

**}**

**else** **if** **(**inicio **>=** final**)**

**return** **-**1**;**

**}**

**Ejecución de la aplicación**

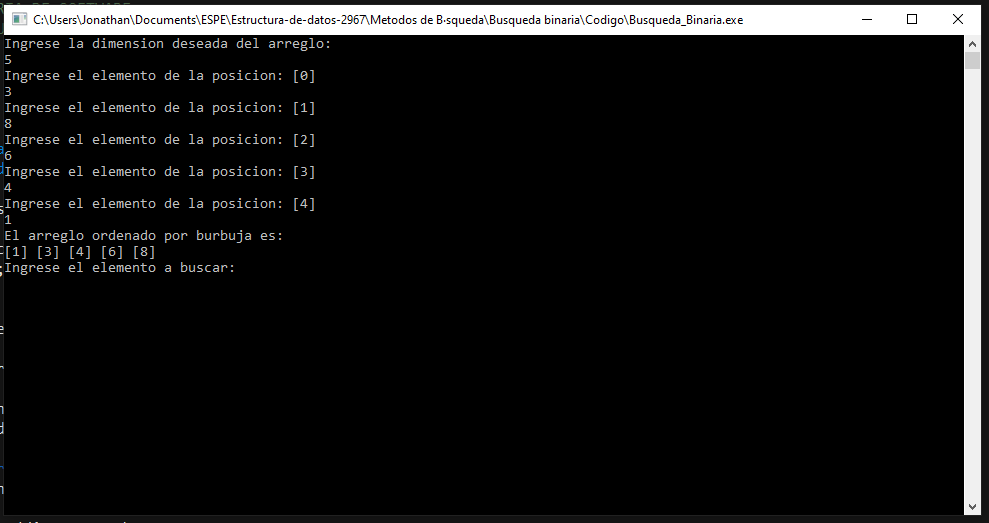
****

Figura : Ingreso de datos por parte del usuario

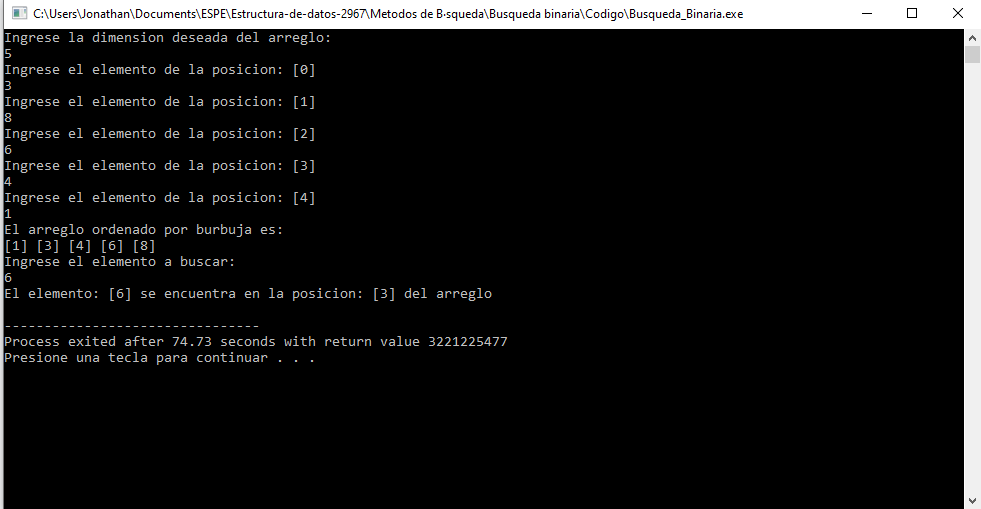
****

Figura : Resultado esperado para el usuario

1. **Método de búsqueda secuencial**

**Objetivo del método secuencial**

Revisar un arreglo definido, elemento por elemento hasta encontrar el elemento que está buscando o hasta llegar al final del arreglo.

**Código de la aplicación:**

* A continuación, se muestra el código fuente que se utilizó para construirá la búsqueda secuencial:

class Busqueda **{**

Public**:**

Busqueda**(**int**\*** arrg**,** int n**);**

void busquedaSecuencial**(**int clave**);**

int**\*** getArreglo**(**void**);**

void setArreglo**(**int**\*** newArreglo**);**

int getTamanio**(**void**);**

void setTamanio**(**int newTamanio**);**

protected**:**

private**:**

int**\*** arreglo**;**

int tamanio**;**

**};**

* Construcción de nuestros prototipos que fueron definidos en la clase Busqueda.h.

#include<iostream>

#include "Busqueda.h"

**using** **namespace** std**;**

Busqueda**::**Busqueda**(**int**\*** arrg**,** int n**)** **{**

**this->**arreglo **=** arrg**;**

**this->**tamanio **=** n**;**

**}**

void Busqueda**::**busquedaSecuencial**(**int clave**)** **{**

bool encontrado **=** **true;**

//Buscar clave o dato en el arreglo.

**for** **(**int j **=** 0**;** j **<** tamanio**;** j**++)** **{**

**if** **(\*(**arreglo **+** j**)** **==** clave**)** **{**

cout **<<** "Se encontro el " **<<** clave **<<** " en la posicion [" **<<** j **+** 1 **<<** "]" **<<** endl**;**

encontrado **=** **false;**

**}**

**}**

**delete[]** arreglo**;**

**if** **(**encontrado**)**

cout **<<** "No se encontro el dato" **<<** endl**;**

**}**

int**\*** Busqueda**::**getArreglo**(**void**)** **{**

**return** arreglo**;**

**}**

void Busqueda**::**setArreglo**(**int**\*** newArreglo**)** **{**

arreglo **=** newArreglo**;**

**}**

int Busqueda**::**getTamanio**(**void**)** **{**

**return** tamanio**;}**

void Busqueda**::**setTamanio**(**int newTamanio**)** **{**

tamanio **=** newTamanio**;**

**}**

* A continuacion se muestra el main del programa

#include<iostream>

#include<stdlib.h>

#include <conio.h>

#include <iostream>

#include "Busqueda.h"

#include "Ingreso.h"

**using** **namespace** std**;**

int main**()** **{**

int clave**,** tamanio**,** num**;**

int **\*** arreglo**;**

Ingreso ingresar**;**

cout **<<** "Busqueda Secuencial" **<<** endl**;**

tamanio **=** ingresar**.**ingresarInt**(**"Ingrese la dimension del arreglo: "**);**

cout **<<** endl**;**

arreglo **=** **(**int **\*)** malloc**(**tamanio **\*** **sizeof** **(**int**));**

//Ingreso de datos al arreglo

**for** **(**int i **=** 0**;** i **<** tamanio**;** i**++)** **{**

cout **<<** "\nIngrese dato [" **<<** i **+** 1 **<<** "]: "**;**

**(\*(**arreglo **+** i**))** **=** ingresar**.**ingresarInt**(**""**);**

**}**

clave **=** ingresar**.**ingresarInt**(**"Ingrese el numero que desea buscar: "**);**

cout **<<** endl**;**

Busqueda buscar **=** Busqueda**(**arreglo**,** tamanio**);**

buscar**.**busquedaSecuencial**(**clave**);**

system**(**"pause"**);**

**return** 0**;**

**}**

**Ejecución del programa:**

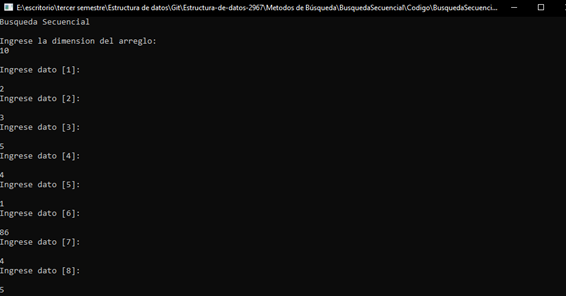
****

Figura 1. ejecución de la aplicación y ingreso de datos.

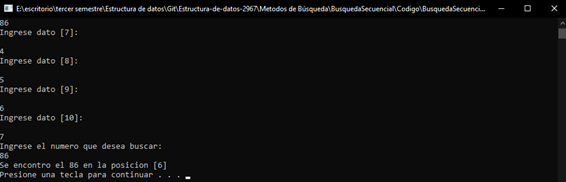


Figura 2. Ingreso de un número que se desea buscar

1. **Puzzle deslizante**

**Objetivo del programa:**

Deslizar los datos de uno en uno, colocarlos ordenadamente en una distribución establecida de antemano (ordenándolos del 1 al número de piezas, alfabéticamente si se utilizan letras, restableciendo el orden que se espera.

**Código:**

* A Continuación se presentan las clases y librería necesarias para el correcto funcionamiento del puzzle deslizante:

#include <iostream>

#include <ctype.h>

#include <string.h>

**using** **namespace** std**;**

class Validacion **{**

public**:**

bool validar**(**string**,** int**);**

**};**

bool Validacion**::**validar**(**string entrada**,** int tipo**)** **{**

**if** **(**tipo **==** 3**)** **{**

**return** **false;**

**}**

int contador **=** 0**;**

**try** **{**

**for** **(**int i **=** 0**;** i **<** entrada**.**length**();** i**++)** **{**

**if** **(**isalpha**(**entrada**[**i**]))** **{**

**throw** 1**;**

**}**

**if** **(!**isdigit**(**entrada**[**i**])** **&&** tipo **==** 1**)** **{**

**throw** 1**;**

**}**

**if** **(**entrada**[**i**]** **==** '.'**)** **{**

contador**++;**

**}**

**if** **((**isdigit**(**entrada**[**i**])** **==** 0 **&&** entrada**[**i**]** **!=** '.'**)** **||** **(**contador **>** 1**))** **{**

**throw** 1**;**

**}**

**}**

**}** **catch** **(**int e**)** **{**

**return** **true;**

**}**

**return** **false;**

**}**

* Metodo que funciona para definir el tablero del puzzle deslizante.

class Tablero **{**

public**:**

Tablero**(**int filas**,** int columnas**);**

**~**Tablero**();**

void inicializar**(**int filas**,** int columnas**);**

void mover**(**int filaActual**,** int columnaActual**,** int filaNueva**,** int columnaNueva**);**

void llenar**(**int filas**,** int columnas**);**

void desordenar**(**int filas**,** int columnas**);**

void imprimir**(**int filas**,** int columnas**);**

bool verificar**(**int filas**,** int columnas**);**

int buscarColumna**(**int filas**,** int columnas**);**

int buscarFila**(**int filas**,** int columnas**);**

int**\*\*** getMatriz**(**void**);**

void setMatriz**(**int**\*\*** newMatriz**);**

protected**:**

private**:**

int**\*\*** matriz**;**

**};**

* La clase Ingreso.h sirve para validar los ingresos por teclado del ususario

#include <iostream>

#include <string>

#include "Validacion.h"

**using** **namespace** std**;**

class Ingreso **{**

public**:**

string leer**(**string**,**int**);**

**};**

string Ingreso**::**leer**(**string mensage**,**int tipo**)** **{**

Validacion validacion**;**

string entrada**;**

cout **<<** mensage **<<** endl**;**

getline**(**cin**,**entrada**);**

**while** **(**validacion**.**validar**(**entrada**,** tipo**))** **{**

cout **<<** "Valor no valido reingrese" **<<** endl**;**

cin **>>** entrada**;**

**}**

**return** entrada**;**

**}**

* Implementación de los prototipos declarados en tablero.h

#include "Tablero.h"

#include <iostream>

#include <time.h>

#include <stdlib.h>

**using** **namespace** std**;**

void Tablero**::**inicializar**(**int filas**,** int columnas**)** **{**

matriz **=** **(**int**\*\*)** malloc**(**filas **\*** **sizeof** **(**int**));**

srand**(**time**(NULL));**

**for** **(**int i **=** 0**;** i **<** columnas**;** i**++)** **{**

**(\*(**matriz **+** i**))** **=** **(**int**\*)** malloc**(**columnas **\*** **sizeof** **(**int**));**

**}**

**}**

void Tablero**::**mover**(**int filaActual**,** int columnaActual**,** int filaNueva**,** int columnaNueva**)** **{**

int aux**;**

aux **=** **\*(\*(**matriz **+** filaActual**)** **+** columnaActual**);**

**\*(\*(**matriz **+** filaActual**)** **+** columnaActual**)** **=** **\*(\*(**matriz **+** filaNueva**)** **+** columnaNueva**);**

**\*(\*(**matriz **+** filaNueva**)** **+** columnaNueva**)** **=** aux**;**

**}**

void Tablero**::**llenar**(**int filas**,** int columnas**)** **{**

int numeros **=** 1**;**

**for** **(**int i **=** 0**;** i **<** filas**;** i**++)** **{**

**for** **(**int j **=** 0**;** j **<** columnas**;** j**++)** **{**

**\*(\*(**matriz **+** i**)** **+** j**)** **=** numeros**;**

numeros**++;**

**}**

**}**

**\*(\*(**matriz **+** filas **-** 1**)** **+** columnas **-** 1**)** **=** **-**1**;**

**}**

void Tablero**::**desordenar**(**int filas**,** int columnas**)** **{**

int filaNueva**;**

int columnaNueva**;**

int auxiliar**;**

srand**(**time**(NULL));**

**for** **(**int i **=** 0**;** i **<** filas**;** i**++)** **{**

**for** **(**int j **=** 0**;** j **<** columnas**;** j**++)** **{**

filaNueva **=** rand**()** **%** **(**filas**);**

columnaNueva **=** rand**()** **%** **(**columnas**);**

auxiliar **=** **(\*(\*(**matriz **+** i**)** **+** j**));**

**\*(\*(**matriz **+** i**)** **+** j**)** **=** **\*(\*(**matriz **+** filaNueva**)** **+** columnaNueva**);**

**(\*(\*(**matriz **+** filaNueva**)** **+** columnaNueva**))** **=** auxiliar**;**

**}**

**}**

**}**

int**\*\*** Tablero**::**getMatriz**(**void**)** **{**

**return** matriz**;**

**}**

void Tablero**::**setMatriz**(**int**\*\*** newMatriz**)** **{**

matriz **=** newMatriz**;**

**}**

Tablero**::**Tablero**(**int filas**,** int columnas**)** **{**

inicializar**(**filas**,** columnas**);**

llenar**(**filas**,** columnas**);**

desordenar**(**filas**,** columnas**);**

**}**

Tablero**::~**Tablero**()** **{**

free**(**matriz**);**

**}**

bool Tablero**::**verificar**(**int filas**,** int columnas**)** **{**

int numero **=** 1**;**

**for** **(**int i **=** 0**;** i **<** filas**;** i**++)** **{**

**for** **(**int j **=** 0**;** j **<** columnas**;** j**++)** **{**

**if** **(\*(\*(**matriz **+** i**)** **+** j**)** **!=** numero**)** **{**

**return** **false;**

**}**

numero**++;**

**}**

**}**

**return** **true;**

**}**

int Tablero**::**buscarFila**(**int filas**,** int columnas**)** **{**

**for** **(**int i **=** 0**;** i **<** filas**;** i**++)** **{**

**for** **(**int j **=** 0**;** j **<** columnas**;** j**++)** **{**

**if** **(\*(\*(**matriz **+** i**)** **+** j**)** **==** **-**1**)** **{**

**return** i**;**

**}**

**}**

**}**

**}**

int Tablero**::**buscarColumna**(**int filas**,** int columnas**)** **{**

**for** **(**int i **=** 0**;** i **<** filas**;** i**++)** **{**

**for** **(**int j **=** 0**;** j **<** columnas**;** j**++)** **{**

**if** **(\*(\*(**matriz **+** i**)** **+** j**)** **==** **-**1**)** **{**

**return** j**;**

**}**

**}**

**}**

**}**

void Tablero**::**imprimir**(**int filas**,** int columnas**)** **{**

system**(**"cls"**);**

cout **<<** endl **<<** endl**;**

**for** **(**int i **=** 0**;** i **<** filas**;** i**++)** **{**

cout **<<** "\t\t"**;**

**for** **(**int j **=** 0**;** j **<** columnas**;** j**++)** **{**

**if** **((\*(\*(**matriz **+** i**)** **+** j**))** **!=** **-**1**)** **{**

cout **<<** **\*(\*(**matriz **+** i**)** **+** j**)** **<<** "\t"**;**

**}** **else** **{**

cout **<<** " \t"**;**

**}**

**}**

cout **<<** endl**;**

**}**

**}**

* Se muestra el main que sirve para ejecutar el aplicativo

#include<iostream>

#include<stdlib.h>

#include <conio.h>

#include <iostream>

#include "Busqueda.h"

#include "Ingreso.h"

**using** **namespace** std**;**

int main**()** **{**

int clave**,** tamanio**,** num**;**

int **\*** arreglo**;**

Ingreso ingresar**;**

cout **<<** "Busqueda Secuencial" **<<** endl**;**

tamanio **=** ingresar**.**ingresarInt**(**"Ingrese la dimension del arreglo: "**);**

cout **<<** endl**;**

arreglo **=** **(**int **\*)** malloc**(**tamanio **\*** **sizeof** **(**int**));**

//Ingreso de datos al arreglo

**for** **(**int i **=** 0**;** i **<** tamanio**;** i**++)** **{**

cout **<<** "\nIngrese dato [" **<<** i **+** 1 **<<** "]: "**;**

**(\*(**arreglo **+** i**))** **=** ingresar**.**ingresarInt**(**""**);**

**}**

clave **=** ingresar**.**ingresarInt**(**"Ingrese el numero que desea buscar: "**);**

cout **<<** endl**;**

Busqueda buscar **=** Busqueda**(**arreglo**,** tamanio**);**

buscar**.**busquedaSecuencial**(**clave**);**

system**(**"pause"**);**

**return** 0**;**

**}**

**Ejecución de programa:**

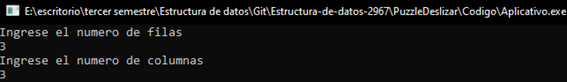
****

Figura 1. Ingreso de la dimensión de la matriz.

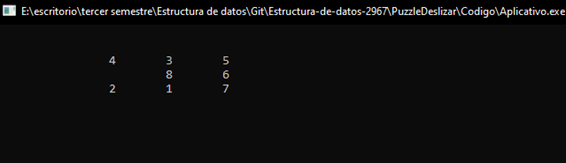


Figura 2. Se genera la matriz con los respectivos números

1. **Contar Vocales**

**Objetivo**

A partir de una cadena ingresada por el usuario u al obtener el dato del nombre, el programa obtendra la cantidad de vocales q se encuentran en dicha cadena.

**Código:**

* A Continuación se presentan las clases y librería necesarias para el correcto funcionamiento:

#include <stdio.h>

#include <stdlib.h>

#include<string.h>

#include "Ingreso.h"

#include "ContadorVocales.h"

int main**()** **{**

//declaracion de las clases Ingreso y lectura

Ingreso lectura**;**

ContadorVocales calcular**;**

int contador**;**

char **\*** cadena**=**lectura**.**leer**(**"Ingrese oracion: "**);**

strupr**(**cadena**);** //comversion de de cadena de minuscula a mayuscula

contador**=**calcular**.**contarVocales**(**cadena**);**//llamando al metodo de la calse ContadorVocales

printf**(**"\n\nHay %i Vocales \n\n"**,** contador**);**

free**(**cadena**);** //liberando la memoria

system**(**"pause"**);**

**return** 0**;**

**}**

* Metodo cuyo proposito es implementar los metodos generados en ContadorVocales.h

#include "ContadorVocales.h"

int ContadorVocales**::**contarVocales**(**char **\*** cadena**)**

**{**

**if(\***cadena**==**'\0'**)**

**{**

**return** 0**;**

**}**

**else**

**{**

**switch(\***cadena**)**

**{**

**case** 'A'**:**

**case** 'E'**:**

**case** 'I'**:**

**case** 'O'**:**

**case** 'U'**:**

cadena**++;**

**return** 1**+**contarVocales**(**cadena**);**

**}**

cadena**++;**

**return** contarVocales**(**cadena**);**

**}**

**}**

**Ejecución de programa:**

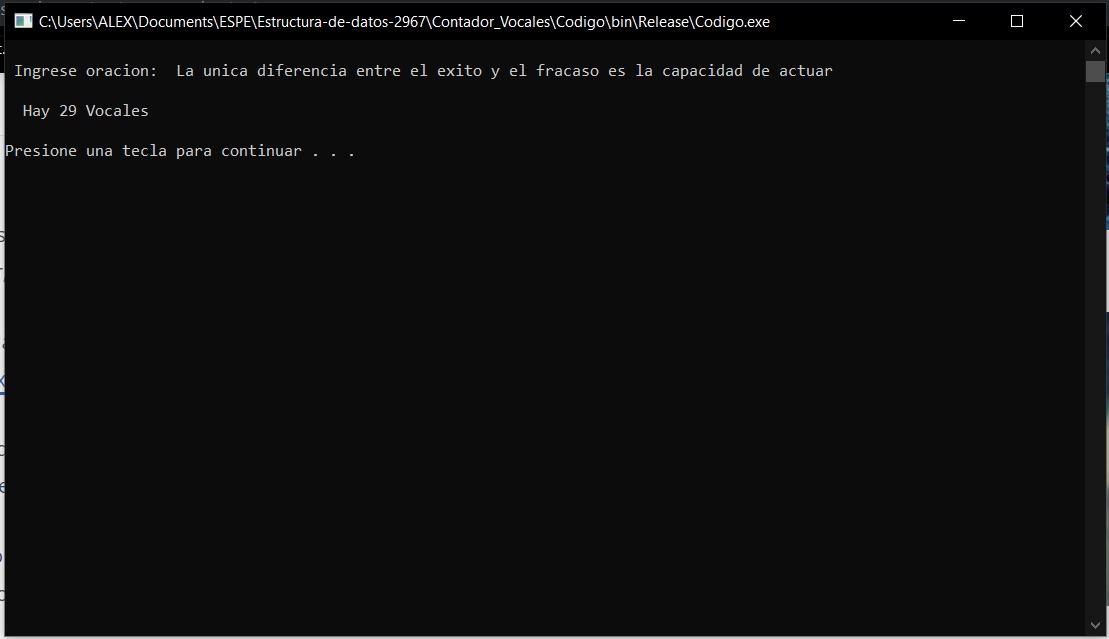


Figura . Ejecución de código

1. **Corrección de la prueba**

**Objetivo:**

Obtener el resultado de la Multiplicación de las matrices según el exponente que se ingrese.

**Código:**

* Implementación de las diferentes clases y funciones para el correcto funcionamiento de la multiplicación de matrices.

class Matriz

**{**

public**:**

int**\*\*** getmatriz**(**void**);**

void setmatriz**(**int**\*\*** newMatriz**);**

int getfilas**(**void**);**

void setfilas**(**int newFilas**);**

int getexp**(**void**);**

void setexp**(**int newExp**);**

int getcolumnas**(**void**);**

void setcolumnas**(**int newColumnas**);**

Matriz**();**

void crear**(**void**);**

void llenar**(**void**);**

void imprimir**(**void**);**

void multiplicar**(**int exponente**);**

protected**:**

private**:**

int filas**;**

int columnas**;**

int exp**;**

int**\*\*** matriz**;**

**};**

* Para validar los datos que ingresa en la matriz.

#include <iostream>

#include <string>

#include <stdlib.h>

**using** **namespace** std**;**

class Ingreso**{**

public**:**

float ingresarFloat**(**char**\*);**

bool validarFloat**(**string**);**

int ingresarInt**(**char**\*);**

bool validarInt**(**string**);**

string ingresarString**(**char**\*);**

bool validarString**(**string**);**

string ingresar10Digitos**(**char**\*);**

bool validar10Digitos**(**string**);**

**};**

float Ingreso**::**ingresarFloat**(**char**\*** msg**){**

string flotante**;**

bool flag**;**

**do{**

**try{**

cout**<<**msg**;**

getline**(**cin**,**flotante**);**

flag **=** validarFloat**(**flotante**);**

**if(**flag**){**

**throw** flotante**;**

**}**

**}catch(**string e**){**

cout**<<**"Dato invalido "**<<**e**<<**endl**;**

**}**

**}while(**flag**);**

**return** atof**(**flotante**.**c\_str**());**

**}**

bool Ingreso**::**validarFloat**(**string valor**){**

bool flag**;**

**for(**int i **=** 0**;** i**<**valor**.**length**();** i**++){**

**if(**isdigit**(**valor**[**i**])){**

flag **=** **false;**

**}else{**

**if(**valor**[**i**]==** '.'**){**

flag **=** **false;**

**}else{**

flag **=** **true;**

**break;**

**}**

**}**

**}**

**return** flag**;**

**}**

int Ingreso**::**ingresarInt**(**char**\*** msg**){**

string numero**;**

bool flag**;**

**do{**

**try{**

cout**<<**msg**;**

getline**(**cin**,**numero**);**

flag **=** validarInt**(**numero**);**

**if(**flag**){**

**throw** numero**;**

**}**

**}catch(**string e**){**

cout**<<**"Dato invalido "**<<**e**<<**endl**;**

**}**

**}while(**flag**);**

**return** atoi**(**numero**.**c\_str**());**

**}**

bool Ingreso**::**validarInt**(**string valor**){**

bool flag**;**

**for(**int i **=** 0**;** i**<**valor**.**length**();** i**++){**

**if(**isdigit**(**valor**[**i**])){**

flag **=** **false;**

**}else{**

**return** **true;**

**}**

**}**

**return** flag**;**

**}**

string Ingreso**::**ingresarString**(**char**\*** msg**){**

string dato**;**

bool flag**;**

**do{**

**try{**

cout**<<**msg**;**

getline**(**cin**,**dato**);**

flag **=** validarString**(**dato**);**

**if(**flag**){**

**throw** dato**;**

**}**

**}catch(**string e**){**

cout**<<**"Dato invalido "**<<**e**<<**endl**;**

**}**

**}while(**flag**);**

**return** dato**;**

**}**

bool Ingreso**::**validarString**(**string valor**){**

bool flag**;**

**for(**int i **=** 0**;** i**<**valor**.**length**();** i**++){**

**if(**isalpha**(**valor**[**i**])** **||** valor**[**i**]** **==** 32**){**

flag **=** **false;**

**}else{**

**return** **true;**

**}**

**}**

**return** flag**;**

**}**

string Ingreso**::**ingresar10Digitos**(**char**\*** msg**){**

string dato**;**

bool flag**;**

**do{**

**try{**

cout**<<**msg**;**

getline**(**cin**,**dato**);**

flag **=** validar10Digitos**(**dato**);**

**if(**flag**){**

**throw** dato**;**

**}**

**}catch(**string e**){**

cout**<<**"Dato invalido "**<<**e**<<**endl**;**

**}**

**}while(**flag**);**

**return** dato**;**

**}**

bool Ingreso**::**validar10Digitos**(**string valor**){**

bool flag**;**

int cont **=** 0**;**

**for(**int i **=** 0**;** i**<**valor**.**length**();** i**++){**

**if(**isdigit**(**valor**[**i**])){**

cont**++;**

**if(**cont **==** 10**){**

flag **=** **false;**

**}else{**

flag **=** **true;**

**}**

**}else{**

**return** **true;**

**}**

**}**

**return** flag**;**

**}**

* Construcción de las funciones declaradas en la matriz.h para utilizar en el main.

#include "Matriz.h"

#include <stdio.h>

#include <iostream>

#include <time.h>

#include <stdlib.h>

int**\*\*** Matriz**::**getmatriz**(**void**)**

**{**

**return** matriz**;**

**}**

void Matriz**::**setmatriz**(**int**\*\*** newMatriz**)**

**{**

matriz **=** newMatriz**;**

**}**

int Matriz**::**getfilas**(**void**)**

**{**

**return** filas**;**

**}**

void Matriz**::**setfilas**(**int newFilas**)**

**{**

filas **=** newFilas**;**

**}**

int Matriz**::**getexp**(**void**)**

**{**

**return** filas**;**

**}**

void Matriz**::**setexp**(**int newExp**)**

**{**

exp **=** newExp**;**

**}**

int Matriz**::**getcolumnas**(**void**)**

**{**

**return** columnas**;**

**}**

void Matriz**::**setcolumnas**(**int newColumnas**)**

**{**

columnas **=** newColumnas**;**

**}**

Matriz**::**Matriz**()**

**{**

**}**

void Matriz**::**crear**(**void**)**

**{**

matriz **=(**int **\*\*)**calloc**(**filas**,sizeof(**int **\*)\***filas**);**

**for(**int j**=**0**;**j**<**columnas**;**j**++)**

**\*(**matriz**+**j**)=(**int **\*)**calloc**(**columnas**,sizeof(**int**\*)\***columnas**);**

**}**

void Matriz**::**llenar**(**void**)**

**{**

srand**(**time**(NULL));**

**for(**int i**=**0**;**i**<**filas**;**i**++)**

**for(**int j**=**0**;**j**<**columnas**;**j**++){**

**\*(\*(**matriz**+**i**)+**j**)=**1**;**//1+rand()%9;

**}**

**}**

void Matriz**::**imprimir**(**void**)**

**{**

**for(**int i**=**0**;**i**<**filas**;**i**++){**

**for(**int j**=**0**;**j**<**columnas**;**j**++)**

**{**

printf**(**"%5d"**,\*(\*(**matriz**+**i**)+**j**));**

**}**

printf**(**"\n"**);**

**}**

**}**

void Matriz**::**multiplicar**(**int exponente**)**

**{**

int**\*\*** mT1**;**

mT1 **=(**int **\*\*)**calloc**(**filas**,sizeof(**int **\*)\***filas**);**

**for(**int j**=**0**;**j**<**columnas**;**j**++){**

**\*(**mT1**+**j**)=(**int **\*)**calloc**(**columnas**,sizeof(**int**\*)\***columnas**);**

**}**

**for(**int i**=**0**;**i**<**filas**;**i**++)**

**for(**int j**=**0**;**j**<**columnas**;**j**++){**

**\*(\*(**mT1**+**i**)+**j**)=\*(\*(**matriz**+**i**)+**j**);**

**}**

**for(**int k**=**1**;**k**<**exponente**;**k**++){**

int **\*\*** mT3**;**

mT3 **=(**int **\*\*)**calloc**(**filas**,sizeof(**int **\*)\***filas**);**

**for(**int j**=**0**;**j**<**columnas**;**j**++)**

**\*(**mT3**+**j**)=(**int **\*)**calloc**(**columnas**,sizeof(**int**\*)\***columnas**);**

**for(**int i**=**0**;**i**<**filas**;**i**++){**

**for(**int j**=**0**;**j**<**filas**;**j**++){**

**for(**int h**=**0**;**h**<**filas**;**h**++){**

**\*(\*(**mT3**+**i**)+**j**)=\*(\*(**mT3**+**i**)+**j**)+(\*(\*(**mT1**+**i**)+**h**))\*(\*(\*(**matriz**+**h**)+**j**));**

**}**

**}**

**}**

matriz**=**mT3**;**

**}**

**}**

* Función principal en la se declara las clases para poder utilizar todas las funciones para poder hacer la multiplicación de matrices.

#include <stdio.h>

#include <iostream>

#include <time.h>

#include <stdlib.h>

#include "Matriz.cpp"

#include "ingreso.h"

**using** **namespace** std**;**

int main**(){**

main**:**

int row**,**col**,**exp**;**

Matriz p**;**

Ingreso lee**;**

row**=** lee**.**ingresarInt**(**"Ingrese # filas: "**);**

p**.**setfilas**(**row**);**

col**=** lee**.**ingresarInt**(**"Ingrese # columnas: "**);**

p**.**setfilas**(**col**);**

cout**<<**endl**;**

**if(**row**>**10**&&**row**>**10**||**row**!=**col**){**

cout**<<**"la matriz tiene q ser cuadrada y su dimenion menor o igual q 10\n"**<<**endl**;**

**}else{**

p**.**setcolumnas**(**col**);**

p**.**setfilas**(**row**);**

p**.**crear**();**

p**.**llenar**();**

p**.**imprimir**();**

exp**=** lee**.**ingresarInt**(**"Ingrese el exponente: "**);**

p**.**setexp**(**exp**);**

cout**<<**endl**;**

p**.**multiplicar**(**p**.**getexp**());**

cout**<<**"matriz elevada "**<<**endl**;**

p**.**imprimir**();**

**}**

system**(**"pause"**);**

system**(**"cls"**);**

**goto** main**;**

**return** 0**;**

**}**

**Ejecución:**

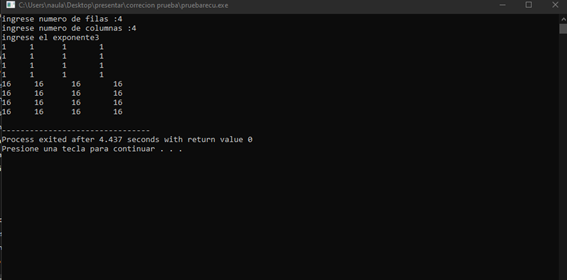


Figura 1. Ejecución de código

1. **Metodo de ordenamiento por Inserción**

**Descripcion**

Es una manera muy natural de ordenar para un ser humano, y puede usarse fácilmente para ordenar un mazo de cartas numeradas en forma arbitraria. Requiere O(n²) operaciones para ordenar una lista de n elementos.

Inicialmente se tiene un solo elemento, que obviamente es un conjunto ordenado. Después, cuando hay *k* elementos ordenados de menor a mayor, se toma el elemento *k+1* y se compara con todos los elementos ya ordenados, deteniéndose cuando se encuentra un elemento menor (todos los elementos mayores han sido desplazados una posición a la derecha) o cuando ya no se encuentran elementos (todos los elementos fueron desplazados y este es el más pequeño). En este punto se *inserta* el elemento *k+1* debiendo desplazarse los demás elementos.

**Objetivo de la aplicación**

Ingreso de una serie de números que decida el usuario para que la aplicación devuelve estos números ordenados.

**Código de la aplicación**

* La librería “Ingreso.h” es la que permite al usuario dar el valor de las dimensiones del cubo mágico.

#include <iostream>

#include <stdlib.h>

#include <string.h>

**using** **namespace** std**;**

class Ingreso **{**

public**:**

int ingresarEntero**();**

double ingresarDouble**();**

float ingresarFloat**();**

string ingresarLetra**();**

bool validar**(**string**);**

bool validarEntero**(**string**);**

bool validarLetra**(**string**);**

**};**

int Ingreso**::**ingresarEntero**()** **{**

string numero**;**

bool valido **=** **false;**

**while** **(!**valido**)** **{**

**try** **{**

getline**(**cin**,** numero**);**

valido **=** validarEntero**(**numero**);**

**if** **(!**valido**)** **{**

**throw** numero**;**

**}**

**}** **catch** **(**string e**)** **{**

cout **<<** "El numero " **<<** e **<<** " no es valido" **<<** endl**;**

**}**

**}**

**return** atoi**(**numero**.**c\_str**());**

**}**

double Ingreso**::**ingresarDouble**()** **{**

string numero**;**

bool valido **=** **false;**

**while** **(!**valido**)** **{**

**try** **{**

getline**(**cin**,** numero**);**

valido **=** validar**(**numero**);**

**if** **(!**valido**)** **{**

**throw** numero**;**

**}**

**}** **catch** **(**string e**)** **{**

cout **<<** "El numero " **<<** e **<<** " no es valido" **<<** endl**;**

**}**

**}**

**return** atof**(**numero**.**c\_str**());**

**}**

float Ingreso**::**ingresarFloat**()** **{**

string numero**;**

bool valido **=** **false;**

**while** **(!**valido**)** **{**

**try** **{**

getline**(**cin**,** numero**);**

valido **=** validar**(**numero**);**

**if** **(!**valido**)** **{**

**throw** numero**;**

**}**

**}** **catch** **(**string e**)** **{**

cout **<<** "El numero " **<<** e **<<** " no es valido" **<<** endl**;**

**}**

**}**

**return** atof**(**numero**.**c\_str**());**

**}**

string Ingreso**::**ingresarLetra**()** **{**

string palabra**;**

bool valido **=** **false;**

**while** **(!**valido**)** **{**

**try** **{**

getline**(**cin**,** palabra**);**

valido **=** validarLetra**(**palabra**);**

**if** **(!**valido**)** **{**

**throw** palabra**;**

**}**

**}** **catch** **(**string e**)** **{**

cout **<<** "La palabra " **<<** e **<<** " no es valida" **<<** endl**;**

**}**

**}**

**return** palabra**;**

**}**

bool Ingreso**::**validar**(**string numero**)** **{**

int inicio **=** 0**;**

**if** **(**numero**.**length**()** **==** 0**)** **{**

**return** 0**;**

**}**

**if** **(**numero**[**0**]** **==** '+' **||** numero**[**0**]** **==** '-'**)** **{**

inicio **=** 1**;**

**if** **(**numero**.**length**()** **==** 1**)** **{**

**return** 0**;**

**}**

**}**

**for** **(**int i **=** inicio**;** i **<** numero**.**length**();** i**++)** **{**

**if** **(!**isdigit**(**numero**[**i**])** **&&** numero**[**i**]** **!=** '.'**)** **{**

**return** 0**;**

**}**

**}**

**return** 1**;**

**}**

bool Ingreso**::**validarEntero**(**string numero**)** **{**

int inicio **=** 0**;**

**if** **(**numero**.**length**()** **==** 0**)** **{**

**return** 0**;**

**}**

**if** **(**numero**[**0**]** **==** '+' **||** numero**[**0**]** **==** '-'**)** **{**

inicio **=** 1**;**

**if** **(**numero**.**length**()** **==** 1**)** **{**

**return** 0**;**

**}**

**}**

**for** **(**int i **=** inicio**;** i **<** numero**.**length**();** i**++)** **{**

**if** **(!**isdigit**(**numero**[**i**]))** **{**

**return** 0**;**

**}**

**}**

**return** 1**;**

**}**

bool Ingreso**::**validarLetra**(**string palabra**)** **{**

char c**;**

**for** **(**int i **=** 0**;** i **<** palabra**.**size**();** i**++)** **{**

c **=** palabra**[**i**];**

**if** **(**isalpha**(**c**)** **==** 0**)** **{**

**if** **(**isspace**(**c**)** **==** 0**)** **{**

**return** 0**;**

**}**

**}**

**}**

**return** 1**;**

**}**

* La librería “recursivo.h” es la que ordena los números que ingresamos

#include <stdio.h>

#include <stdlib.h>

class Insercion **{**

private**:**

int **\***vector**;**

public**:**

int**\*** inicializar**(**int**);**

int**\*** encerar**(**int**,** int**\*);**

int**\*** ingresaNumeros**(**int**,** int**\*);**

void whileRecursivo**(**int**,** int**,** int**\*);**

void insercionRecursivo**(**int**,** int**\*);**

void mostrar**(**int**,** int**\*);**

**};**

int**\*** Insercion**::**inicializar**(**int tamanio**)** **{**

vector **=** **(**int**\*)** malloc**(**tamanio **\*** **sizeof** **(**int**));**

**return** vector**;**

**}**

int**\*** Insercion**::**encerar**(**int tamanio**,** int **\***vector**)** **{**

**for** **(**int i **=** 0**;** i **<** tamanio**;** i**++)** **{**

**\*(**vector **+** i**)** **=** 0**;**

**}**

**return** vector**;**

**}**

int**\*** Insercion**::**ingresaNumeros**(**int tamanio**,** int **\***vector**)** **{**

vector **=** inicializar**(**tamanio**);**

vector **=** encerar**(**tamanio**,** vector**);**

**for** **(**int i **=** 0**;** i **<** tamanio**;** i**++)** **{**

scanf**(**"%d"**,** vector **+** i**);**

**}**

**return** vector**;**

**}**

void Insercion**::**whileRecursivo**(**int fin**,** int j**,** int **\***vector**)** **{**

**while** **(**j **>=** 0 **&&** **\*(**vector **+** j**)** **>** fin**)** **{**

**\*(**vector **+** **(**j **+** 1**))** **=** **\*(**vector **+** j**);**

j**--;**

**}**

**\*(**vector **+** **(**j **+** 1**))** **=** fin**;**

**}**

void Insercion**::**insercionRecursivo**(**int n**,** int**\*** vector**)** **{**

**if** **(**n **<=** 1**)** **{**

**return;**

**}**

insercionRecursivo**(**n **-** 1**,** vector**);**

int fin **=** **\*(**vector **+** **(**n **-** 1**));**

int j **=** n **-** 2**;**

whileRecursivo**(**fin**,** j**,** vector**);**

**}**

void Insercion**::**mostrar**(**int tamanio**,** int **\***vector**)** **{**

**for** **(**int i **=** 0**;** i **<** tamanio**;** i**++)** **{**

printf**(**"%d "**,** **\*(**vector **+** i**));**

**}**

**}**

* A continuacion se muestra el main del programa

#include <stdio.h>

#include "recursivo.h"

#include "ingreso.h"

int main**()** **{**

Ingreso leer**;**

Insercion insercion**;**

int **\***vector**,** tamanio**;**

printf**(**"Ingrese el tamanio del vector: "**);**

tamanio **=** leer**.**ingresarEntero**();**

printf**(**"Ingrese los datos del vector: "**);**

vector **=** insercion**.**ingresaNumeros**(**tamanio**,** vector**);**

insercion**.**mostrar**(**tamanio**,** vector**);**

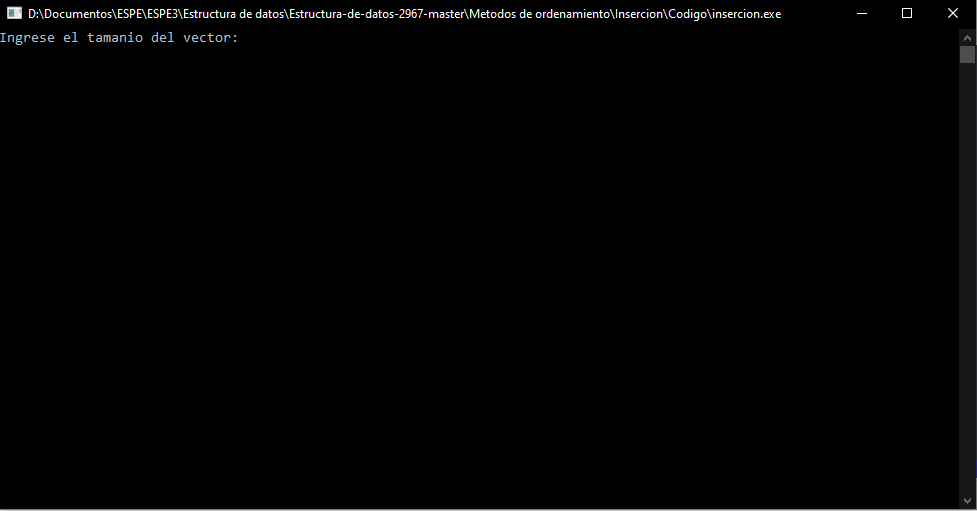
printf**(**"\nEl vector ordenado es: \n"**);**

insercion**.**insercionRecursivo**(**tamanio**,** vector**);**

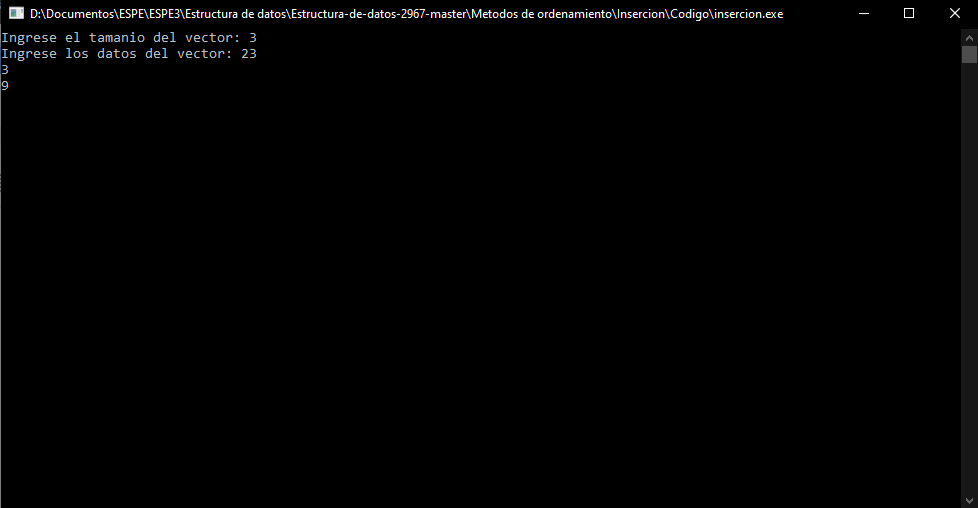
insercion**.**mostrar**(**tamanio**,** vector**);**

**}**

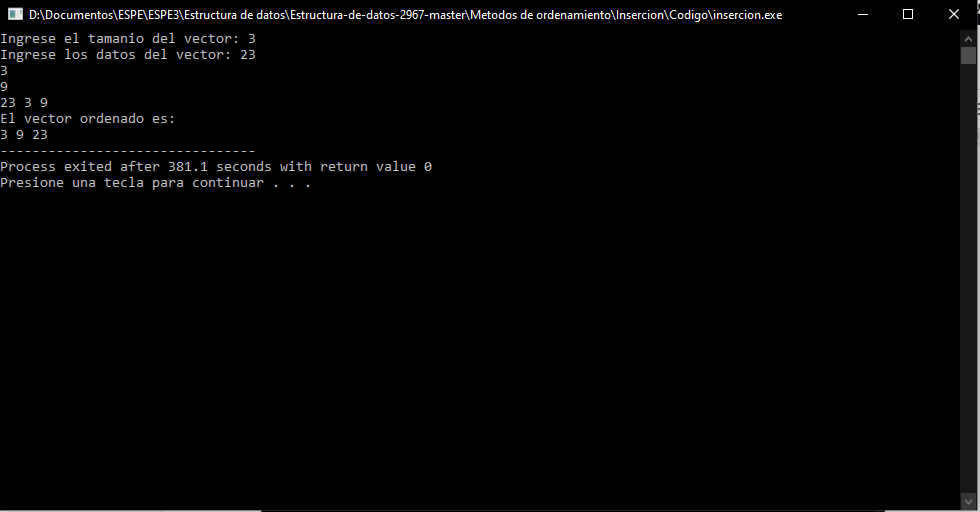
**Ejecución del programa**

****

**Figura 1. Ingreso del tamaño del vector**

****

**Figura 2. Ingreso de los números para ordenar**

****

**Figura 3. Números ordenados**

1. **Ocho reinas**

**Objetivo:**

Generar diferentes tipos de soluciones posibles para las ocho o menos reinas en un tablero de ajedrez.

**Códigod el programa:**

* se generan los prototipos de funciones, necesarias para el correcto funcionamiento de la aplicación.

#include <iostream>

#include <fstream>

**using** **namespace** std**;**

class Reina

**{**

public**:**

Reina**(**int n1**);**

**~**Reina**();**

void solucion**(**int x**,** int y**,** int n1**);**

void bloquear**(**int x**,** int y**);**

void quitarRelleno**(**int x**,** int y**);**

void mostrar**();**

void solucionReinas**();**

protected**:**

private**:**

bool**\*\*** validar**;**

char**\*\*** tablero**;**

fstream enter**;**

int contador**;**

int n**;**

**};**

* Construir los prototipos que fueron declarados en Reina.h

#include "Reina.h"

#include <iostream>

#include <fstream>

Reina**::**Reina**(**int n1**)**

**{**

n**=**n1**;**

validar **=** **new** bool **\*[**n**];**

tablero**=** **new** char **\*[**n**];**

**for(**int i**=**0**;**i**<**n**;**i**++){**

validar**[**i**]=new** bool **[**n**];**

tablero**[**i**]=new** char**[**n**];**

**for(**int j**=**0**;**j**<**n**;**j**++){**

**\*(\*(**validar**+**i**)+**j**)=false;**

**\*(\*(**tablero**+**i**)+**j**)=**'.'**;**

**}**

**}**

enter**.**open**(**"solucion.txt"**,**fstream**::**out**);** //para leer in, para salir es out escribir

enter**<<**"solucion "**<<**n**<<**"\*"**<<**n**<<**endl**<<**endl**;**

contador**=**0**;**

**}**

Reina**::~**Reina**()**

**{**

// TODO : implement

**}**

void Reina**::**solucion**(**int x**,** int y**,** int n1**)**

**{**

**\*(\*(**tablero**+**x**)+**y**)=**'R'**;**

bloquear**(**x**,**y**);**

**if(**n1**==**n**){**

mostrar**();**

**}else{**

**for(**int i**=**0**;**i**<**n**;**i**++){**

**if(\*(\*(**validar**+**i**)+**y**+**1**)==false){**

solucion**(**i**,**y**+**1**,**n1**+**1**);**

**}**

**}**

**}**

quitarRelleno**(**x**,**y**);**

**}**

void Reina**::**bloquear**(**int x**,** int y**)**

**{**

int aux1**,**aux2**;**

aux2**=**y**;**

aux1**=**0**;**

**while(**aux1**<**n**){** //vertical

**\*(\*(**validar**+**aux1**)+**aux2**)=true;**

aux1**++;**

**}**

aux2**=**0**;**

aux1**=**x**;**

**while(**aux2**<**n**){**

**\*(\*(**validar**+**aux1**)+**aux2**)=true;**

aux2**++;**

**}**

aux2**=**y**;** //Diagonales

aux1**=**x**;**

**while(**aux1**>**0**&&**aux2**>**0**){**

aux1**--;**

aux2**--;**

**}**

aux2**=**y**;**

aux1**=**x**;**

**while(**aux1**<**n**&&**aux2**>**0**){**

aux1**++;**

aux2**--;**

**}**

aux1**--;**

aux2**++;**

**while(**aux1**>=**0**&&**aux2**<**n**){**

**\*(\*(**validar**+**aux1**)+**aux2**)=true;**

aux1**--;**

aux2**++;**

**}**

**}**

void Reina**::**quitarRelleno**(**int x**,** int y**)**

**{**

**\*(\*(**tablero**+**x**)+**y**)=**'.'**;**

**for(**int i**=**0**;**i**<**n**;**i**++){**

**for(**int j**=**0**;**j**<**n**;**j**++){**

**\*(\*(**validar**+**i**)+**j**)=false;**

**}**

**}**

**for(**int i**=**0**;**i**<**n**;**i**++){**

**for(**int j**=**0**;**j**<**n**;**j**++){**

**if(\*(\*(**tablero**+**i**)+**j**)==**'R'**)**

bloquear**(**i**,**j**);**

**}**

**}**

**}**

void Reina**::**mostrar**()**

**{**

contador**++;**

enter**<<**"Solucion N "**<<**contador**<<**endl**;**

**for(**int i**=**0**;**i**<**n**;**i**++){**

**for(**int j**=**0**;**j**<**n**;**j**++){**

enter**<<\*(\*(**tablero**+**i**)+**j**)<<**" "**;**

**}**

enter**<<**endl**;**

**}**

enter**<<**endl**;**

**}**

void Reina**::**solucionReinas**()**

**{**

**for(**int i**=**0**;**i**<**n**;**i**++){**

solucion**(**i**,**0**,**1**);**

**}**

enter**.**close**();**

**}**

* sirve para validar la entrada de datos al programa según el tipo que se necesite.

#include <iostream>

#include <string>

**using** **namespace** std**;**

class Ingreso**{**

public**:**

string ingresar**(**string**);**

bool validarTipoFloat**(**string**);**

bool validarTipoInt**(**string**);**

bool validarTipoString**(**string**);**

**};**

string Ingreso**::**ingresar**(**string msg**){**

string dato\_a\_validar**;**

bool esValido **=** **false;**

**while(!**esValido**){**

**try{**

cout **<<** msg**;**

getline**(**cin**,**dato\_a\_validar**);**

**if(**msg**.**find**(**"flotante"**)!=** std**::**string**::**npos**){**

esValido **=** validarTipoFloat**(**dato\_a\_validar**);**

**}**

**else** **if(**msg**.**find**(**"entero"**)!=** std**::**string**::**npos**){**

esValido **=** validarTipoInt**(**dato\_a\_validar**);**

**}**

**else** **if(**msg**.**find**(**"cadena"**)!=** std**::**string**::**npos**){**

esValido **=** validarTipoString**(**dato\_a\_validar**);**

**}**

**if(!**esValido**){**

**throw** dato\_a\_validar**;**

**}**

**}catch(**string e**){**

cout **<<** "El dato (" **<<** e **<<** ") no es valido" **<<** endl**;**

**}**

**}**

**return** dato\_a\_validar**;**

**}**

bool Ingreso**::**validarTipoFloat**(**string numero**){**

int inicio **=** 0**;**

**if(**numero**.**length**()** **==** 0**){**

**return** 0**;**

**}**

**if(**numero**[**0**]** **==** '+' **||** numero**[**0**]** **==** '-'**){**

inicio **=** 1**;**

**if(**numero**.**length**()** **==** 1**){**

**return** 0**;**

**}**

**}**

**for(**int i **=** inicio**;** i**<**numero**.**length**();** i**++)** **{**

**if** **(!**isdigit**(**numero**[**i**])** **&&** numero**[**i**]** **!=** '.'**)** **{**

**return** 0**;**

**}**

**}**

**return** 1**;**

**}**

bool Ingreso**::**validarTipoInt**(**string numero**){**

int inicio **=** 0**;**

**if(**numero**.**length**()** **==** 0**){**

**return** 0**;**

**}**

**if(**numero**[**0**]** **==** '+' **||** numero**[**0**]** **==** '-'**){**

inicio **=** 1**;**

**if(**numero**.**length**()** **==** 1**){**

**return** 0**;**

**}**

**}**

**for(**int i **=** inicio**;** i**<**numero**.**length**();** i**++)** **{**

**if** **(!**isdigit**(**numero**[**i**]))** **{**

**return** 0**;**

**}**

**}**

**return** 1**;**

**}**

bool Ingreso**::**validarTipoString**(**string numero**){**

**if(**numero**.**length**()** **==** 0**){**

**return** 0**;**

**}**

**for(**int i **=** 0**;** i**<**numero**.**length**();** i**++)** **{**

**if** **(**isdigit**(**numero**[**i**]))** **{**

**return** 0**;**

**}**

**}**

**return** 1**;**

**}**

* Se arma el programa con todas las clases para generar las soluciones de las ocho Reinas.

#include <iostream>

#include <fstream>

#include "Reina.h"

#include "Ingreso.h"

/\* run this program using the console pauser or add your own getch, system("pause") or input loop \*/

**using** **namespace** std**;**

int main**(**int argc**,** char**\*\*** argv**)** **{**

int n**;**

cout**<<**"\tCuantas reinas: "**<<**endl**;**

cin**>>**n**;**

Reina **\***obj**=** **new** Reina**(**n**);**

obj**->**solucionReinas**();**

cout**<<**"Se creo el archivo solucion.txt"**<<**endl**;**

**return** 0**;**

**}**

**Ejecucion de programa:**

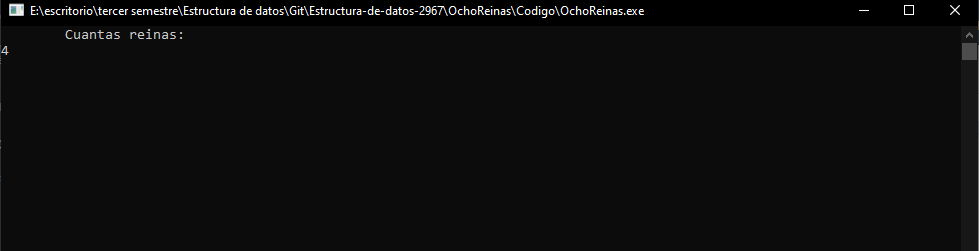
****

Figura 1. ingreso del número de reinas.

* A continuacion se muestra el archivo “soluciones.txt” donde se almacenan todas las soluciones brindadas por backtracking.

solución 4**\***4

Solución N 1

R **.** **.** **.**

**.** R **.** **.**

**.** **.** R **.**

**.** **.** **.** R

Solución N 2

R **.** **.** **.**

**.** **.** **.** R

**.** R **.** **.**

**.** **.** R **.**

Solución N 3

R **.** **.** **.**

**.** **.** R **.**

**.** **.** **.** R

**.** R **.** **.**

Solución N 4

**.** **.** R **.**

R **.** **.** **.**

**.** R **.** **.**

**.** **.** **.** R

Solución N 5

**.** **.** R **.**

R **.** **.** **.**

**.** **.** **.** R

**.** R **.** **.**

Solución N 6

**.** R **.** **.**

**.** **.** R **.**

R **.** **.** **.**

**.** **.** **.** R

Solución N 7

**.** R **.** **.**

**.** **.** **.** R

R **.** **.** **.**

**.** **.** R **.**

1. **Generar e-mail:**

**Objetivo:**

Generar el correo con los datos ingresados por el usuario que son los nombre y apellidos y comprobar si la fecha es palindroma.

**Código:**

* generar los prototipos de las funciones.

#include <string>

#include "Provincia.h"

#include "FechaNacimiento.h"

**using** **namespace** std**;**

class Persona**:** public Provincia**,** public FechaNacimiento

**{**

public**:**

string getCedula**(**void**);**

void setCedula**(**string newCedula**);**

string getNombre**(**void**);**

void setNombre**(**string newNombre**);**

string getApellido**(**void**);**

void setApellido**(**string newApellido**);**

string getNacionalidad**(**void**);**

void setNacionalidad**(**string newNacionalidad**);**

string getGenero**(**void**);**

void setGenero**(**string newGenero**);**

string getEstCivil**(**void**);**

void setEstCivil**(**string newEstCivil**);**

string getTelf**(**void**);**

void setTelf**(**string newTelf**);**

Persona**();**

private**:**

string cedula**;**

string nombre**;**

string apellido**;**

string nacionalidad**;**

string genero**;**

string estCivil**;**

string telf**;**

**};**

string Persona**::**getCedula**(**void**)**

**{**

**return** cedula**;**

**}**

void Persona**::**setCedula**(**string newCedula**)**

**{**

cedula **=** newCedula**;**

**}**

string Persona**::**getNombre**(**void**)**

**{**

**return** nombre**;**

**}**

void Persona**::**setNombre**(**string newNombre**)**

**{**

nombre **=** newNombre**;**

**}**

string Persona**::**getApellido**(**void**)**

**{**

**return** apellido**;**

**}**

void Persona**::**setApellido**(**string newApellido**)**

**{**

apellido **=** newApellido**;**

**}**

string Persona**::**getNacionalidad**(**void**)**

**{**

**return** nacionalidad**;**

**}**

void Persona**::**setNacionalidad**(**string newNacionalidad**)**

**{**

nacionalidad **=** newNacionalidad**;**

**}**

string Persona**::**getGenero**(**void**)**

**{**

**return** genero**;**

**}**

void Persona**::**setGenero**(**string newGenero**)**

**{**

genero **=** newGenero**;**

**}**

string Persona**::**getEstCivil**(**void**)**

**{**

**return** estCivil**;**

**}**

void Persona**::**setEstCivil**(**string newEstCivil**)**

**{**

estCivil **=** newEstCivil**;**

**}**

string Persona**::**getTelf**(**void**)**

**{**

**return** telf**;**

**}**

void Persona**::**setTelf**(**string newTelf**)**

**{**

telf **=** newTelf**;**

**}**

Persona**::**Persona**()**

**{**

**}**

* generar prototipos de las funciones.

#include <string>

**using** **namespace** std**;**

#include "Persona.h"

class Alumno **:** public Persona

**{**

public**:**

string getIdAlumno**(**void**);**

void setIdAlumno**(**string newIdAlumno**);**

string getEmail**(**void**);**

void setEmail**(**string newEmail**);**

Alumno**();**

private**:**

string idAlumno**;**

string email**;**

**};**

string Alumno**::**getIdAlumno**(**void**)**

**{**

**return** idAlumno**;**

**}**

void Alumno**::**setIdAlumno**(**string newIdAlumno**)**

**{**

idAlumno **=** newIdAlumno**;**

**}**

string Alumno**::**getEmail**(**void**)**

**{**

**return** email**;**

**}**

void Alumno**::**setEmail**(**string newEmail**)**

**{**

email **=** newEmail**;**

**}**

Alumno**::**Alumno**()**

**{**

**}**

Provincia**.**h

#include <string>

#include "Canton.h"

**using** **namespace** std**;**

class Canton**;**

class Provincia **:** public Canton

**{**

public**:**

string getIdProvincia**(**void**);**

void setIdProvincia**(**string newIdProvincia**);**

string getNombreProvincia**(**void**);**

void setNombreProvincia**(**string newNombreProvincia**);**

Provincia**();**

protected**:**

string idProvincia**;**

string nombreProvincia**;**

**};**

string Provincia**::**getIdProvincia**(**void**)**

**{**

**return** idProvincia**;**

**}**

void Provincia**::**setIdProvincia**(**string newIdProvincia**)**

**{**

idProvincia **=** newIdProvincia**;**

**}**

string Provincia**::**getNombreProvincia**(**void**)**

**{**

**return** nombreProvincia**;**

**}**

void Provincia**::**setNombreProvincia**(**string newNombreProvincia**)**

**{**

nombreProvincia **=** newNombreProvincia**;**

**}**

Provincia**::**Provincia**()**

**{**

**}**

#include <string>

#include "Parroquia.h"

**using** **namespace** std**;**

class Canton**:** public Parroquia

**{**

public**:**

string getIdCanton**(**void**);**

void setIdCanton**(**string newIdCanton**);**

string getNombreCanton**(**void**);**

void setNombreCanton**(**string newNombreCanton**);**

Canton**();**

protected**:**

string idCanton**;**

string nombreCanton**;**

**};**

string Canton**::**getIdCanton**(**void**)**

**{**

**return** idCanton**;**

**}**

void Canton**::**setIdCanton**(**string newIdCanton**)**

**{**

idCanton **=** newIdCanton**;**

**}**

string Canton**::**getNombreCanton**(**void**)**

**{**

**return** nombreCanton**;**

**}**

void Canton**::**setNombreCanton**(**string newNombreCanton**)**

**{**

nombreCanton**=**newNombreCanton**;**

**}**

Canton**::**Canton**()**

**{**

**}**

#include <string>

**using** **namespace** std**;**

class Parroquia

**{**

public**:**

string getIdParroq**(**void**);**

void setIdParroq**(**string newIdParroq**);**

string getNombreParroq**(**void**);**

void setNombreParroq**(**string newNombreParroq**);**

Parroquia**();**

protected**:**

string idParroq**;**

string nombreParroq**;**

**};**

string Parroquia**::**getIdParroq**(**void**)**

**{**

**return** idParroq**;**

**}**

void Parroquia**::**setIdParroq**(**string newIdParroq**)**

**{**

idParroq **=** newIdParroq**;**

**}**

string Parroquia**::**getNombreParroq**(**void**)**

**{**

**return** nombreParroq**;**

**}**

void Parroquia**::**setNombreParroq**(**string newNombreParroq**)**

**{**

nombreParroq **=** newNombreParroq**;**

**}**

Parroquia**::**Parroquia**()**

**{**

**}**

#include <string>

#include <stdlib.h>

**using** **namespace** std**;**

class FechaNacimiento

**{**

public**:**

int getDia**(**void**);**

void setDia**(**int newDia**);**

int getMes**(**void**);**

void setMes**(**int newMes**);**

int getAnio**(**void**);**

void setAnio**(**int newAnio**);**

FechaNacimiento**();**

bool fechaPalindroma**();**

private**:**

int dia**;**

int mes**;**

int anio**;**

**};**

int FechaNacimiento**::**getDia**(**void**)**

**{**

**return** dia**;**

**}**

void FechaNacimiento**::**setDia**(**int newDia**)**

**{**

dia **=** newDia**;**

**}**

int FechaNacimiento**::**getMes**(**void**)**

**{**

**return** mes**;**

**}**

void FechaNacimiento**::**setMes**(**int newMes**)**

**{**

mes **=** newMes**;**

**}**

int FechaNacimiento**::**getAnio**(**void**)**

**{**

**return** anio**;**

**}**

void FechaNacimiento**::**setAnio**(**int newAnio**)**

**{**

anio **=** newAnio**;**

**}**

FechaNacimiento**::**FechaNacimiento**()**

**{**

**}**

bool FechaNacimiento**::**fechaPalindroma**(){**

string palindromo**;**

char**\*** conv**;**

conv**=(**char**\*)**malloc**(**8**\*sizeof(**char**));**

sprintf**(**conv**,**"%d%d%d"**,**dia**,**mes**,**anio**);**

palindromo**.**append**(**conv**);**

free**(**conv**);**

int x**=**palindromo**.**length**();**

**for(**int j**=**0**;**j**<(**palindromo**.**length**()/**2**);**j**++){**

**if(**palindromo**.**at**(**j**)!=**palindromo**.**at**(**x**-**1**)){**

**return** **false;**

**}**

x**--;**

**}**

**return** **true;**

**}**

#include <string>

**using** **namespace** std**;**

class GenerarEmail

**{**

public**:**

virtual string genEmail**(**string nombre**,** string apellido**)=**0**;**

**};**

#include <iostream>

#include "Excluir.h"

#include <string>

**using** **namespace** std**;**

class Ingreso**{**

public**:**

string ingresarString**(**char**\*** **);**

int ingresar**(**char**\*** **);**

string ingresarStringN**(**char**\*** **);**

**};**

string Ingreso**::**ingresarString**(**char **\***msg**){**

string valor**;**

bool flag**;**

Excluir comprobante**;**

cout**<<**msg**<<**endl**;**

**do{**

cin**>>**valor**;**

flag**=**comprobante**.**controlS**(**valor**);**

**}while(**flag**==false);**

**return** valor**;**

**}**

string Ingreso**::**ingresarStringN**(**char **\***msg**){**

string valor**;**

bool flag**;**

Excluir comprobante**;**

cout**<<**msg**<<**endl**;**

**do{**

cin**>>**valor**;**

flag**=**comprobante**.**controlSN**(**valor**);**

**}while(**flag**==false);**

**return** valor**;**

**}**

int Ingreso**::**ingresar**(**char **\***msg**){**

string valor**;**

bool flag**;**

Excluir comprobante**;**

cout**<<**msg**<<**endl**;**

**do{**

cin**>>**valor**;**

flag**=**comprobante**.**control**(**valor**);**

**}while(**flag**==false);**

int num**;**

num **=** strtof**((**valor**).**c\_str**(),**0**);**

**return** num**;**

**}**

#include <string>

#include <stdio.h>

#include <cstdlib>

#include "GenerarEmail.h"

**using** **namespace** std**;**

class PruebaEmail **:** public GenerarEmail

**{**

public**:**

string genEmail**(**string nombre**,** string apellido**);**

PruebaEmail**();**

**};**

string PruebaEmail**::**genEmail**(**string nombre**,** string apellido**)**

**{**

string email**,** min**;**

char**\*** alocator**;**

alocator**=(**char**\*)**malloc**(**1**\*sizeof(**char**));**

int x**=**nombre**.**find**(**" "**);**

**if(**x**>**1**){**

sprintf**(**alocator**,**"%c%c"**,**nombre**.**at**(**0**),**nombre**.**at**(**nombre**.**find**(**" "**)+**1**));**

**}**

**else{**

sprintf**(**alocator**,**"%c"**,**nombre**.**at**(**0**));**

**}**

email**.**append**(**alocator**);**

**if(**apellido**.**find**(**" "**)>**0**){**

email**.**append**(**apellido**.**substr**(**0**,**apellido**.**find**(**" "**)).**c\_str**());**

**}**

**else{**

email**.**append**(**apellido**.**c\_str**());**

**}**

**for(**int i **=** 0**;** i**<**email**.**length**();**i**++){**

int y**=(**int**)**email**.**at**(**i**);**

char c**=**tolower**(**y**);**

char**\*** minus**;**

minus**=(**char**\*)**malloc**(**1**\*sizeof(**char**));**

sprintf**(**minus**,**"%c"**,**c**);**

min**.**append**(**minus**);**

free**(**minus**);**

**}**

**return** min**;**

**}**

PruebaEmail**::**PruebaEmail**()**

**{**

**}**

#include <iostream>

#include <string>

#include <bits/stdc++.h>

#include <stdlib.h>

**using** **namespace** std**;**

class Excluir**{**

public**:**

bool control**(**string s**){**

int d**=**0**;**

**for(**int i**=**0**;**i**<**s**.**size**();**i**++)**

**{**

**if(!**isdigit**(**s**[**i**])){**

cout**<<**"Error"**<<**endl**;**

**return** **false;**

**}**

**}**

**return** **true;**

**}**

bool controlS**(**string s**){**

int d**=**0**;**

**for(**int i**=**0**;**i**<**s**.**size**();**i**++)**

**{**

**if(**isdigit**(**s**[**i**])&&**s**[**i**]!=**' '**){**

cout**<<**"Error"**<<**endl**;**

**return** **false;**

**}**

**}**

**return** **true;**

**}**

bool controlSN**(**string s**){**

int d**=**0**;**

**for(**int i**=**0**;**i**<**s**.**size**();**i**++)**

**{**

**if(!**isdigit**(**s**[**i**])){**

cout**<<**"Error"**<<**endl**;**

**return** **false;**

**}**

**}**

**return** **true;**

**}**

**};**

* construcción del programa con las funciones ya declaradas.

#include <stdio.h>

#include <iostream>

#include <string>

#include <fstream>

#include "Alumno.h"

#include "PruebaEmail.h"

#include "Ingreso.h"

int main**(){**

Ingreso ex**;**

string idPa**,**idPr**,**idCa**,** idAlumno**,** nombre**,**apellido**;**

Alumno alum **=** Alumno**();**

PruebaEmail pru **=** PruebaEmail**();**

cout**<<**"Ingrese su nombre"**<<**endl**;**

getline**(**cin**,**nombre**);**

alum**.**setNombre**(**nombre**);**

cout**<<**"Ingrese su apellido"**<<**endl**;**

getline**(**cin**,**apellido**);**

alum**.**setApellido**(**apellido**);**

alum**.**setCedula**(**ex**.**ingresarStringN**(**"Ingrese su cedula"**));**

alum**.**setEstCivil**(**ex**.**ingresarString**(**"Ingrese su estado civil"**));**

alum**.**setGenero**(**ex**.**ingresarString**(**"Ingrese su genero"**));**

alum**.**setTelf**(**ex**.**ingresarStringN**(**"Ingrese su telefono"**));**

alum**.**setNacionalidad**(**ex**.**ingresarString**(**"Ingrese su nacionalidad"**));**

alum**.**setAnio**(**ex**.**ingresar**(**"Ingrese su anio de nacimiento"**));**

alum**.**setMes**(**ex**.**ingresar**(**"Ingrese su mes de nacimiento"**));**

**while(**alum**.**getMes**()>**12**){**

alum**.**setMes**(**ex**.**ingresar**(**"Ingrese su mes de nacimiento"**));**

**}**

alum**.**setDia**(**ex**.**ingresar**(**"Ingrese su dia de nacimiento"**));**

**while(**alum**.**getDia**()>**28**&&**alum**.**getMes**()==**2**){**

alum**.**setDia**(**ex**.**ingresar**(**"Ingrese su dia de nacimiento"**));**

**}**

**while(**alum**.**getDia**()>**31**){**

alum**.**setDia**(**ex**.**ingresar**(**"Ingrese su dia de nacimiento"**));**

**}**

cout**<<**"Ingrese el id de su provincia"**<<**endl**;**

cin**>>**idPr**;**

alum**.**setIdProvincia**(**idPr**);**

alum**.**setNombreProvincia**(**ex**.**ingresarString**(**"Ingrese el nombre de su provincia de residencia"**));**

cout**<<**"Ingrese el id de su canton"**<<**endl**;**

cin**>>**idCa**;**

alum**.**setIdCanton**(**idCa**);**

alum**.**setNombreCanton**(**ex**.**ingresarString**(**"Ingrese el nombre de su canton de residencia"**));**

cout**<<**"Ingrese el id de su parroquia"**<<**endl**;**

cin**>>**idPa**;**

alum**.**setIdParroq**(**idPa**);**

alum**.**setNombreParroq**(**ex**.**ingresarString**(**"Ingrese el nombre de su parroquia de residencia"**));**

cout**<<**"Ingrese su id de alumno"**<<**endl**;**

cin**>>**idAlumno**;**

alum**.**setIdAlumno**(**idAlumno**);**

alum**.**setEmail**(**pru**.**genEmail**(**alum**.**getNombre**(),**alum**.**getApellido**()));**

string line**;**

fstream out**;**

out**.**open**(**"example.txt"**,**fstream**::**app**);**

int i**=**0**;**

ifstream myfile **(**"example.txt"**);**

**if** **(**myfile**.**is\_open**())**

**{**

**while** **(** getline **(**myfile**,**line**)** **)**

**{**

**if(**pru**.**genEmail**(**alum**.**getNombre**(),**alum**.**getApellido**()).**compare**(**line**.**substr**(**0**,**pru**.**genEmail**(**alum**.**getNombre**(),**alum**.**getApellido**()).**length**()))==**0**){**

i**++;**

**}**

**}**

myfile**.**close**();**

line**=**pru**.**genEmail**(**alum**.**getNombre**(),**alum**.**getApellido**());**

**if(**i**!=**0**){**

char**\*** num**;**

num**=(**char**\*)**malloc**(**1**\*sizeof(**char**));**

sprintf**(**num**,**"%d"**,**i**);**

line**.**append**(**num**);**

**}**

out**<<**line**<<**endl**;**

out**.**close**();**

**}**

**else{**

**}**

**if(**alum**.**fechaPalindroma**()){**

cout**<<**"es palindromo"**<<**endl**;**

**}else{**

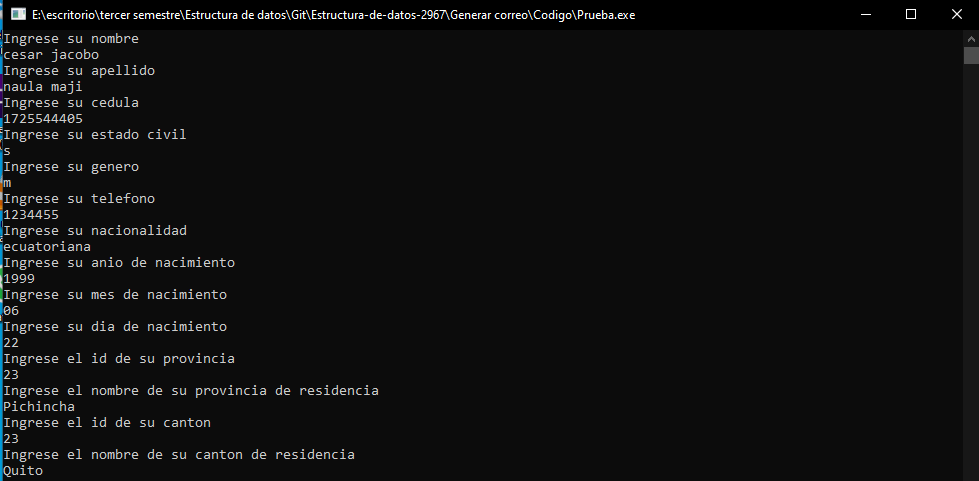
cout**<<**"Nel prro"**<<**endl**;**

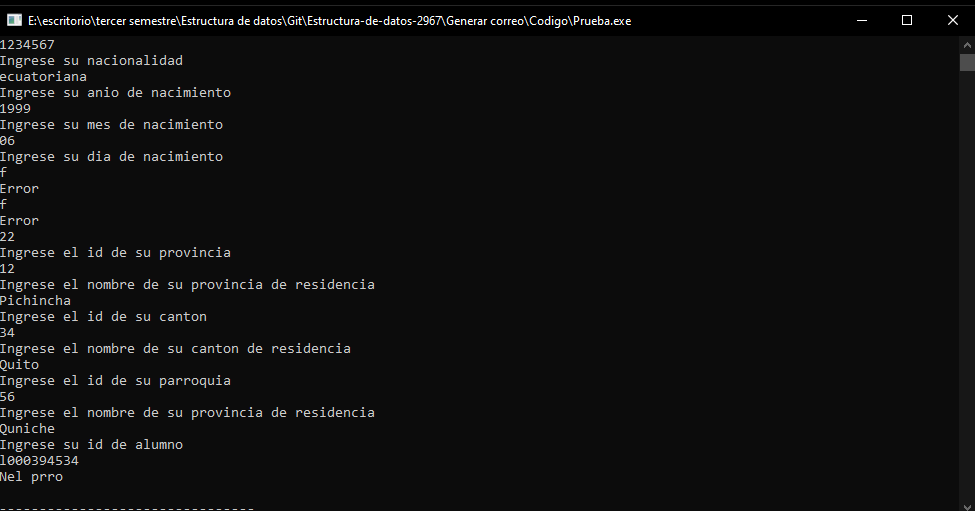
**}**

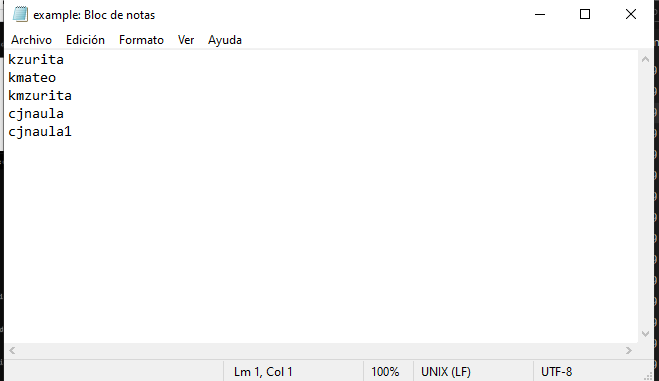
**return** 0**;**

**}**

**Ejecución:**

**** Figura 1. ejecución de programa.

****Figura 2. ingreso de datos

Figura 3. Correos generados.