

# Reporte

Probabilidad y estadística

Integrantes:

Axel Issai Aleman Delgado  
Orlando Samuel Martínez Dorantes  
José Rodolfo Cervantes Cabrera

# Contenido

Introducción.....	1
Probabilidad.....	1
Teorema de bayes.....	1
Conteo.....	2
Permutaciones.....	2
Combinaciones.....	2
Estadística.....	3
Media aritmética.....	3
Mediana.....	3
Moda.....	4
Varianza.....	4
Desviación estándar.....	4
Cuartiles.....	5
Percentiles.....	6
Desarrollo.....	7
Resultados.....	11
Conclusiones.....	17
Apéndice.....	18

# Introducción.

## Probabilidad.

Se refiere al estudio del azar y la incertidumbre en cualquier situación en la cual varios posibles sucesos pueden ocurrir.

La disciplina de la probabilidad proporciona los métodos para cuantificar las oportunidades y probabilidades asociadas con varios sucesos.

### ***Teorema de bayes.***

El teorema de Bayes parte de una situación en la que es posible conocer las probabilidades de que ocurran una serie de sucesos A.

A esta se añade un suceso B cuya ocurrencia proporciona cierta información, porque las probabilidades de ocurrencia de B son distintas según el suceso A que haya ocurrido.

Conociendo que ha ocurrido el suceso B, la fórmula del teorema de Bayes nos indica cómo modifica esta información las probabilidades de los sucesos A.

El teorema se rige mediante esta fórmula:

$$P(A|B) = \frac{P(B|A) \cdot P(A)}{P(B)}$$

Donde:

- $A, B$  eventos
- $P(A|B)$  la probabilidad de A dado B
- $P(B|A)$  la probabilidad de B dado A
- $P(A), P(B)$  las probabilidades independientes de A y B

### **Conteo.**

Las técnicas de conteo son unos métodos matemáticos que permiten saber cuántas combinaciones u opciones distintas se tienen de los elementos dentro de un mismo grupo de objetos.

El conteo es utilizado para determinar el número de posibilidades diferentes que existen al realizar un experimento.

### **Permutaciones.**

Una permutación es la variación del orden o posición de los elementos de un conjunto ordenado o una secuencia finita de ***n*** objetos

Se representa mediante la siguiente fórmula:

$${}_nP_r = \frac{n!}{(n-r)!}$$

Donde:

- $n$  = objetos o eventos
- $r$  = el número de maneras que podemos elegir objetos o eventos

### **Combinaciones.**

Se llaman combinaciones de  $n$  objetos de orden  $r$  a los distintos grupos que se pueden formar al escoger secuencialmente  $r$  objetos de entre  $n$  posibles, de modo cada una de las combinaciones es distinta de las demás, si difiere en uno de sus objetos por lo menos, sin importar el orden.

La fórmula de las combinaciones sin repetición se representa de esta forma:

$$C_{n,r} = \binom{n}{r} = \frac{n!}{r!(n-r)!}$$

Donde:

- $n$  = observaciones totales.
- $r$  = número de elementos.

## Estadística.

La Estadística es una ciencia que trata del recuento, ordenación y clasificación de los datos obtenidos por las observaciones, para poder hacer comparaciones y sacar conclusiones.

La disciplina de estadística nos enseña cómo realizar juicios inteligentes y tomar decisiones informadas en la presencia de incertidumbre y variación.

### **Media aritmética.**

Es el resultado de la suma de sus valores, del conjunto de análisis, entre el número de datos.

Es decir, dados los  $n$  números  $x_1, x_2, \dots, x_n$  la media aritmética se define como:

$$\bar{x} = \frac{1}{n} \sum_{i=1}^n x_i = \frac{x_1 + x_2 + \dots + x_n}{n}$$

Donde:

- $\bar{x}$  = El símbolo de la media aritmética
- $x_1, x_2, \dots, x_n$  = Son conocidos como observaciones.

### **Mediana.**

Es el valor central de una serie de datos ordenados.

Esta es representada por el símbolo:  $Me$ .

Fórmula:

$$Me = L_{i-1} + \frac{\frac{N}{2} - F_{i-1}}{fi} \cdot a$$

Donde:

- $L_{i-1}$  = Límite inferior del intervalo mediana
- $a$  = Amplitud del intervalo mediana
- $F_{i-1}$  = Frecuencia acumulada anterior al intervalo mediana
- $fi$  = Frecuencia absoluta del intervalo mediana
- $N$  = Total de datos

**Moda.**

Es el valor más común, más típico, que ocurre más frecuentemente en un conjunto de datos.

Puede no existir o no estar definida.

Se representa por el símbolo:  $M_o$

**Varianza.**

Es una medida muy sensible de la variabilidad y base de muchas técnicas estadísticas.

Cuenta con dos propiedades importantes:

1. La varianza de una constante es cero
2. Si se tiene la varianza  $\sigma^2$  de un conjunto de datos y a cada observación se multiplica por una constante  $b$ , entonces la nueva varianza de los datos se obtiene multiplicando a la varianza de los datos por  $b^2$

La varianza se describe de la siguiente manera:

$$\sigma^2 = \frac{(x_1 - \bar{X})^2 + (x_2 - \bar{X})^2 + \dots + (x_n - \bar{X})^2}{N}$$

Donde:

- $x_1, x_2, \dots, x_n$  = Las observaciones,
- $\bar{X}$  = La media
- $N$  = El número total de datos

**Desviación estándar.**

Es el promedio de desviación de las puntuaciones con respecto a la media.

Esta medida se expresa en las unidades originales de medición de la distribución.

Cuanto mayor sea la dispersión de los datos alrededor de la media, mayor será la desviación estándar.

Se simboliza con  $\sigma$ .

La desviación estándar se representa matemáticamente de esta forma:

$$\sigma = \sqrt{\frac{\sum_i^N (X_i - \bar{X})^2}{N}}$$

Donde:

- $X_i$  = Las observaciones
- $\bar{X}$  = La media
- $N$  = El número total de datos

### **Cuartiles.**

Los cuartiles son valores que dividen una muestra de datos en cuatro partes iguales. Utilizando cuartiles puede evaluar rápidamente la dispersión y la tendencia central de un conjunto de datos, que son los pasos iniciales importantes para comprender sus datos

Existen 2 formas para calcularlos dependiendo si son o no datos agrupados.

Calculo de cuartiles para datos agrupados:

$$Q_k = L_i + \frac{\frac{kN}{4} - F_{i-1}}{f_i} \cdot a_i \quad k = 1, 2, 3$$

Donde:

- $L_i$  = El límite inferior de la clase donde se encuentra el cuartil
- $N$  = Suma de las frecuencias absolutas
- $F_{i-1}$  = La frecuencia acumulada anterior a la clase del cuartil
- $a_i$  = La amplitud de la clase

Calculo de cuartiles para datos no agrupados:

$$Q_1 = \frac{n+1}{4} \quad Q_2 = \frac{n+1}{2} \quad Q_3 = 3 \frac{n+1}{4}$$

### **Percentiles.**

El percentil es una medida de posición usada en estadística que indica, una vez ordenados los datos de menor a mayor, el valor de la variable por debajo del cual se encuentra un porcentaje dado de observaciones en un grupo. Los percentiles son los 99 valores que dividen una serie de datos ordenados en 100 partes iguales.

Los percentiles dan los valores correspondientes al 1%, al 2%... y al 99% de los datos.

Fórmula para datos no agrupados:

$$\begin{array}{ll} \text{Cuando } n \text{ es par} & \text{cuando } n \text{ es impar} \\ P_k = \frac{k \cdot n}{100} & P_k = \frac{k \cdot (n+1)}{100} \end{array}$$

Donde:

$k$  = Percentil deseado (1, 2, 3, 4, 5, 6, 7, 8, 9,..., 45,...82,... y 99)

$n$  = Total de datos

Fórmula para datos agrupados:

$$P_k = L_k + \frac{k \cdot \left(\frac{N}{100}\right) - F_k}{f_k} \cdot c \quad k = 1, 2, 3, \dots, 99$$

Donde:

- $L_k$  = Límite real inferior de la clase del decil  $k$
- $N$  = Numero de datos
- $F_k$  = La frecuencia acumulada anterior a la clase del decil  $k$
- $f_k$  = frecuencia de la clase del decil  $k$
- $c$  = Longitud del intervalo de la clase del decil  $k$



# Desarrollo.

Especificaciones del código:

El código fue realizado en el lenguaje de programación C; dividimos nuestro programa en diferentes archivos de cabecera donde cada una de estas representa uno de los temas solicitados.

Estos archivos de cabecera son:

- **MenuPrincipal.h**

Dentro de este archivo tenemos 2 funciones, una llamada MenuPrincipal la cual nos proporciona un menú en el cual podremos seleccionar alguno de los 3 temas abordados, funciona por medio de un *switch...case*; al seleccionar alguna de las opciones se hace la llamada a la función y al archivo de cabecera correspondiente, la otra es una función adicional llamada Salida, agregada como mensaje de despedida al salir del programa, en esta se muestra una imagen junto con los nombres de los desarrolladores del programa.

- **Estadistica.h**

En este archivo de cabecera se encuentran múltiples funciones, cada una de ellas es responsable de devolvernos el resultado deseado. Dentro de este archivo tenemos las funciones

MenuEstadistica: Esta es la función principal, se encarga de cargar el archivo de tipo .txt con los valores numéricos que se desean calcular, extraer los datos de dicho archivo y almacenarlos en un arreglo, además en ella se imprimen los resultados devueltos por las demás funciones; por último nos ofrece la posibilidad de grabar dichos resultados en un archivo de texto.

ordenar: Una vez que los datos son extraídos y almacenados en el arreglo se envían a esta función, la cual tiene como propósito ordenar los datos de manera ascendente, ya que posteriormente será requerido para realizar algunas operaciones.

media: Como su nombre lo indica esta función nos devuelve la media o promedio del arreglo generado por la función MenuEstadistica.

mediana: Esta es una de las funciones que requieren los datos ordenados de manera ascendente; aquí disponemos de una estructura de selección *if...else* la cual decide de qué forma se obtendrá la mediana, si el total de datos es un número par lo que hará será devolver el promedio de los dos elementos centrales; en caso contrario, que el total de datos sea un número impar, la función nos devolverá el valor del elemento central.

moda: Esta función está conformada por varios ciclos anidados, los cuales se encargan de evaluar cada uno de los datos y evaluar las veces que dicho dato se repite. Nos devuelve el valor que más repeticiones tiene.

varianza: Esta función cuenta con un ciclo de repetición *for* dentro del cual a cada uno de los elementos del arreglo se les resta el valor de la media y posteriormente se elevan al cuadrado (utilizando la función `pow` disponible en la librería `<math.h>`), los resultados son sumados y almacenados dentro de una variable; por último se divide el valor de la variable entre el total de datos y el resultado es devuelto a la función principal.

deviacion: esta función nos devuelve la raíz cuadrada de la varianza (obtenida por la función *varianza*) haciendo uso de la función *sqrt* (disponible en la librería `<math.h>`).

cuartiles y percentiles: dichas funciones nos proporcionan el valor de los cuartiles y percentiles, respectivamente, solicitados. Para ello, cuenta con un ciclo de selección *if...else* el cual determina qué fórmula se utilizará, si el total de datos es par se utiliza la fórmula  $\frac{k * n}{4}$  para cuartiles, y  $\frac{k * n}{100}$  para percentiles; en caso contrario, que el total de datos sea impar, se utiliza la fórmula  $\frac{k * (n + 1)}{4}$  para cuartiles, y  $\frac{k * (n + 1)}{100}$  para los percentiles. Una vez que se determina la fórmula a utilizar, se aplica y el resultado se almacena en una variable, posteriormente, dicha variable será evaluada utilizando un *if...else*, si el valor de la variable es un entero la función nos devolverá el valor que se encuentra en esa posición dentro de nuestro arreglo. En caso que la variable contenga un valor decimal se interpola multiplicando la parte decimal por la resta del valor posterior menos el valor que se encuentra en la posición entera, por último se le suma el valor de la variable.

- **probabilidad.h**

Este archivo de cabecera solo cuenta con una función, la cual es encargada de solicitar los valores requeridos y realizar todas las operaciones necesarias, así como mostrar en pantalla los resultados de estos; Al igual que en **Estadistica.h** nos ofrece la posibilidad de grabar dichos resultados en un archivo de texto.

- **Conteo.h**

Dentro de este archivo de cabecera disponemos de 4 funciones.

MenuConteo: Es la función principal, nos muestra un pequeño menú en el cual podremos seleccionar alguna de las siguientes 4 opciones: Combinaciones, permutaciones, regresar al menú principal y salir del programa; haciendo uso de un *switch...case* que llama a la función solicitada por el usuario.

factorial: Se encarga de obtener el valor factorial de los valores solicitados.

combinaciones: Cuenta con un ciclo *if...else*, el cual solicita al usuario los valores de  $n$  y  $r$  para posteriormente enviarlos a la función *factorial*, una vez que tengamos los factoriales requeridos se devuelve el resultado de aplicar la fórmula de combinaciones, en caso de que  $r > n$  muestra un mensaje de error y solicita ingresar nuevamente los valores.

permutaciones: hace lo mismo que la función combinaciones, salvo que en esta función se aplica la fórmula de permutaciones.

- **Grabar\_resultados.h**

Este es el encargado de grabar los resultados en archivos de texto, si el usuario así lo solicita. Cuenta con dos funciones: *Guardar\_estadistica* y *Guardar\_probabilidad*.

Guardar\_estadistica: graba los resultados obtenidos por **Estadistica.h** en un archivo de texto.

Guardar\_probabilidad: graba los resultados obtenidos por **probabilidad.h** en un archivo de texto.

Dichos archivos de texto tienen el siguiente formato en su nombre: *NombreTema\_Fecha\_Hora*, (por ejemplo: *Estadistica\_25-07-20\_11-07-08.txt*), dentro de estos estarán los resultados mostrados en pantalla.

- **color.h**

Este es un archivo de cabecera adicional, creado con el fin de proporcionar colores a nuestra ventana así como al texto mostrado en pantalla para hacerlo más llamativo y fácil de entender.

Funcionamiento:

El programa está compuesto por 3 secciones que abordan los temas que estudiamos durante el cuatrimestre los cuales son:

- Estadística
  1. Media
  2. Mediana
  3. Moda
  4. Varianza
  5. Desviación estándar
  6. Cuartiles
  7. Percentiles
- Probabilidad
  1. Probabilidad de eventos
  2. Teorema de bayes
- Conteo
  1. Combinaciones
  2. Permutaciones

El programa funciona mediante un menú en el cual contiene las 3 secciones para elegir.

Posteriormente de escoger la sección deseada abre una ventana donde se le podrá cargar el archivo de tipo .txt con los valores numéricos que se desean calcular.

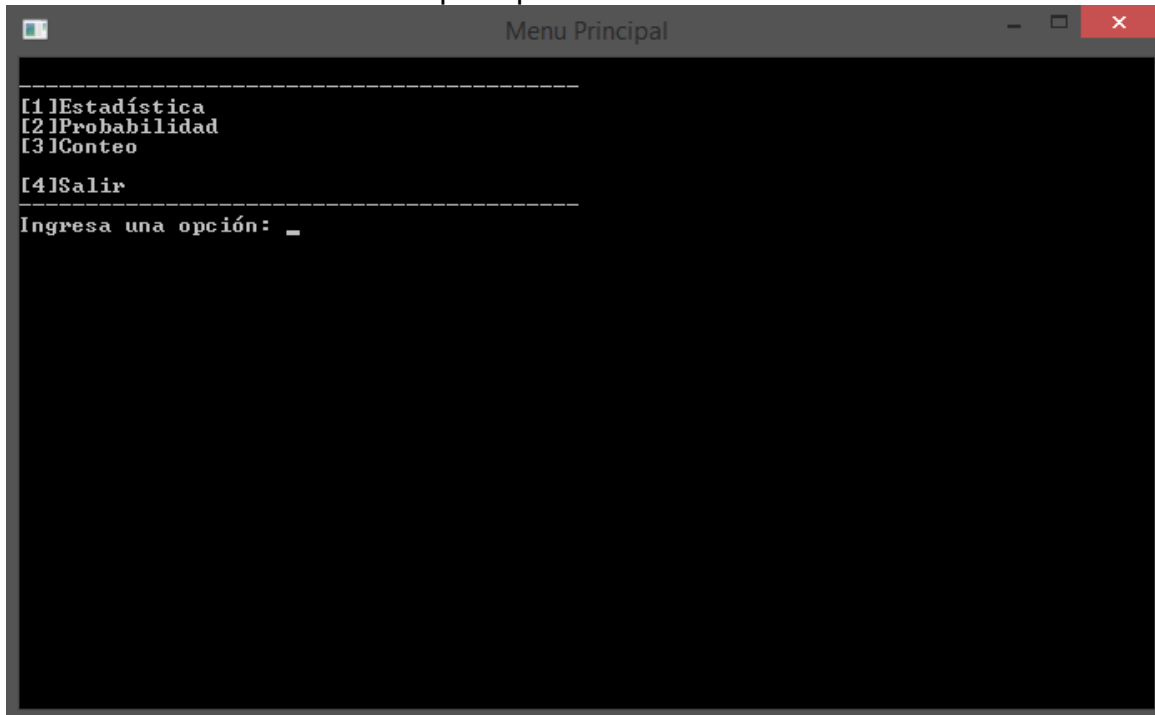
El programa extraerá los datos del archivo de texto y verificara que sean mínimo 30 valores numéricos, si cumple la condición procederá a hacer los cálculos correspondientes.

Por último mostrará los resultados obtenidos en la pantalla.

# Resultados.

Archivo: *MenuPrincipal.h* Función: *MenuPrincipal*

Muestra el menú de selección principal



```
Menu Principal

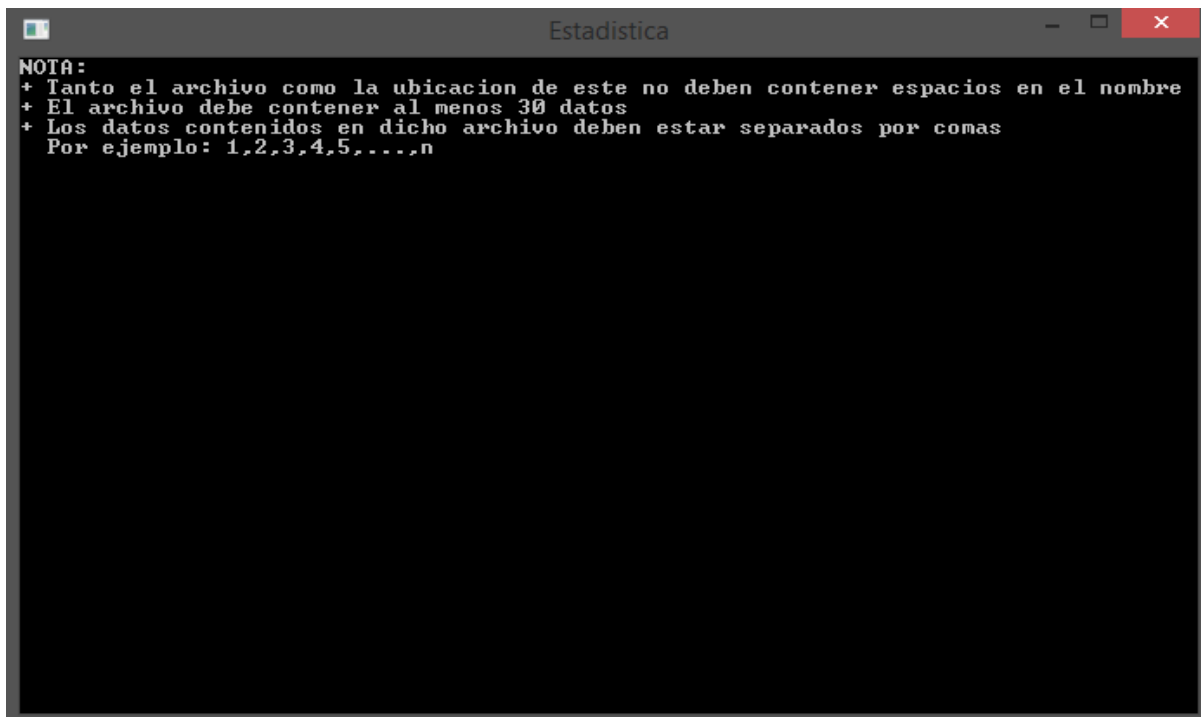
-----
[1] Estadística
[2] Probabilidad
[3] Conteo
[4] Salir
-----
Ingresa una opción: _
```

Archivo: *Estadistica.h*

Función: *MenuEstadisticaPrimero*

muestra

ciertos puntos a tener en cuenta



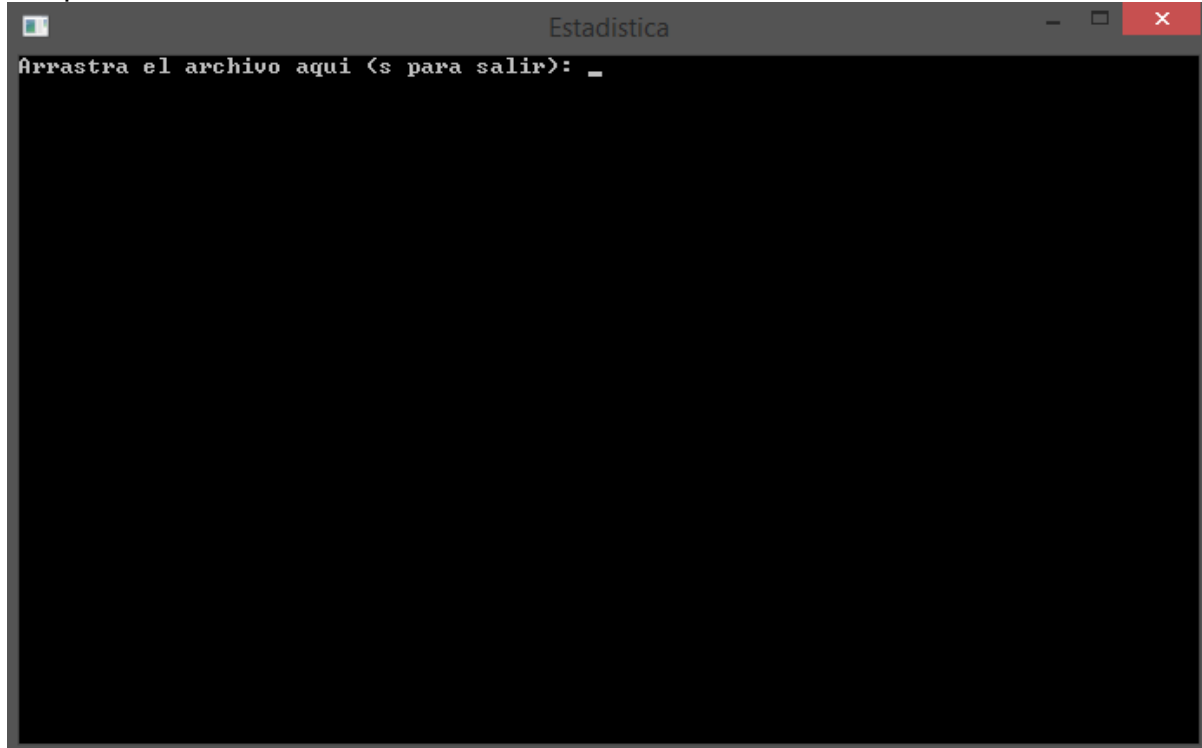
```
Estadistica

NOTA:
+ Tanto el archivo como la ubicacion de este no deben contener espacios en el nombre
+ El archivo debe contener al menos 30 datos
+ Los datos contenidos en dicho archivo deben estar separados por comas
  Por ejemplo: 1,2,3,4,5,....,n
```

Archivo: *Estadistica.h*

Función: *MenuEstadistica*

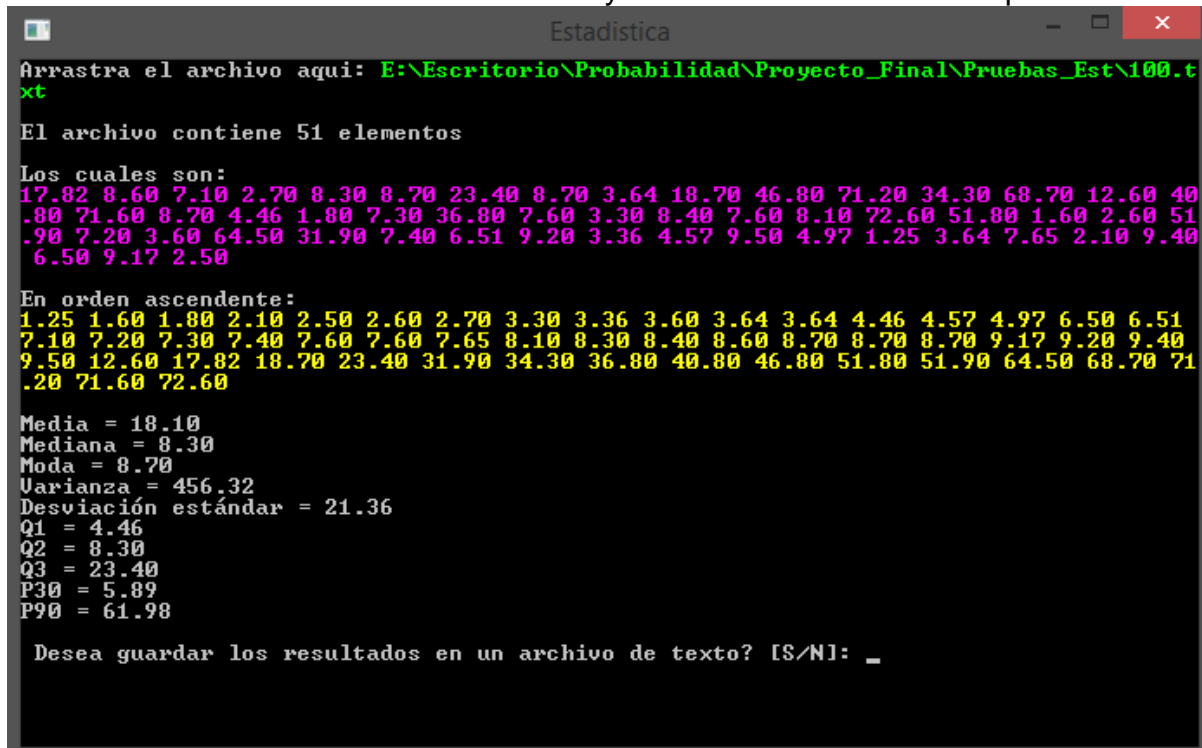
Después nos solicita el archivo txt de donde se extraerán los datos



Archivo: *Estadistica.h*

Función: *MenuEstadistica*

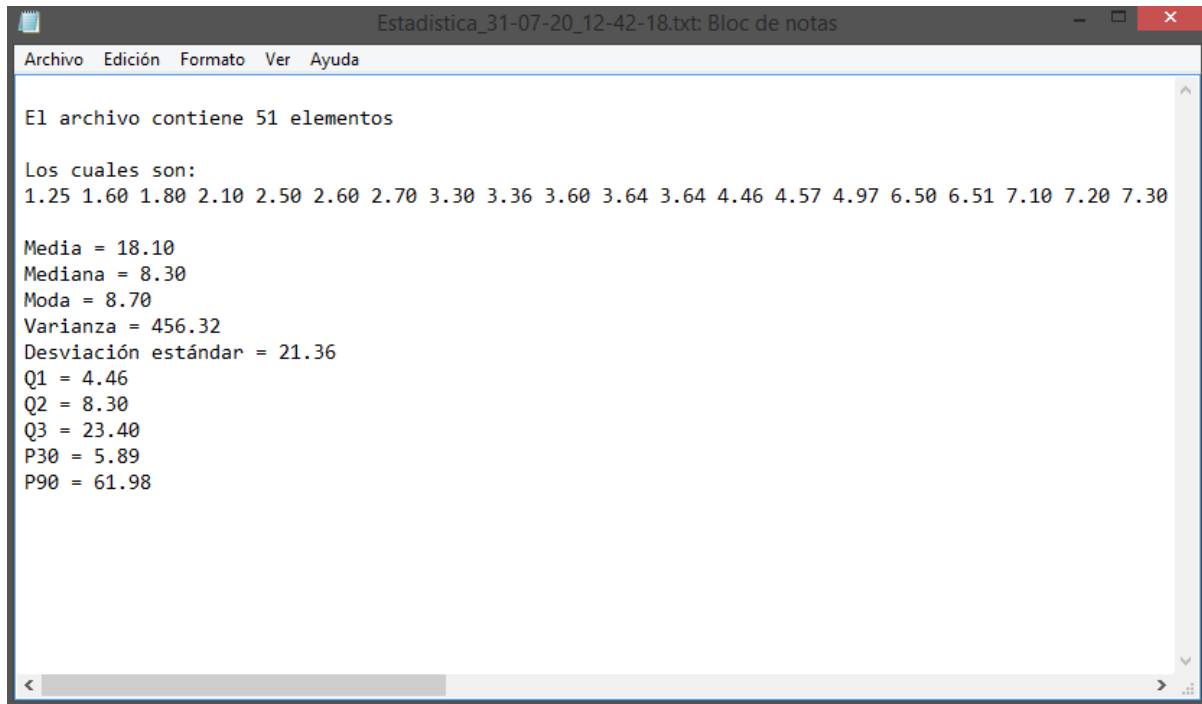
Realiza el llamado a las demás funciones y muestra los resultados en pantalla



Archivo: *Grabar\_resultados.h*

Función: *Guardar\_estadistica*

Graba los resultados obtenidos en un archivo de texto



```
El archivo contiene 51 elementos

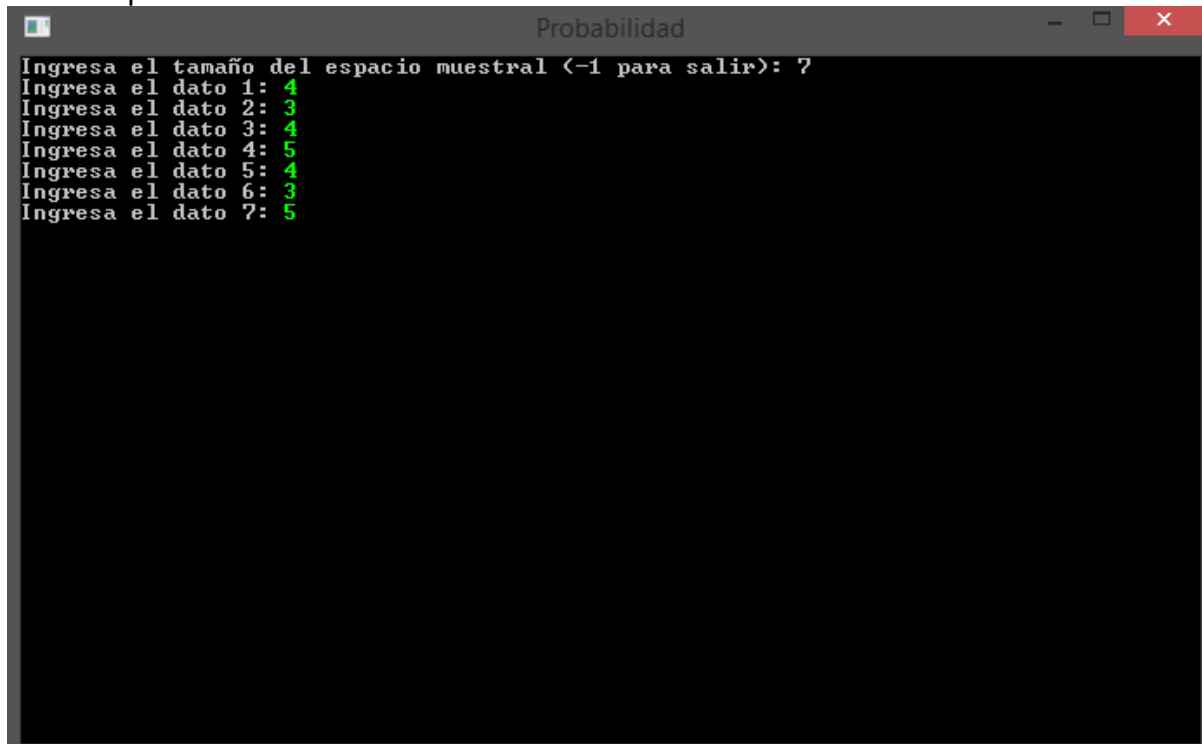
Los cuales son:
1.25 1.60 1.80 2.10 2.50 2.60 2.70 3.30 3.36 3.60 3.64 3.64 4.46 4.57 4.97 6.50 6.51 7.10 7.20 7.30

Media = 18.10
Mediana = 8.30
Moda = 8.70
Varianza = 456.32
Desviación estándar = 21.36
Q1 = 4.46
Q2 = 8.30
Q3 = 23.40
P30 = 5.89
P90 = 61.98
```

Archivo: *probabilidad.h*

Función: *probabilidad*

Primero pide los datos necesarios



```
Ingresar el tamaño del espacio muestral <-1 para salir>: ?
Ingresar el dato 1: 4
Ingresar el dato 2: 3
Ingresar el dato 3: 4
Ingresar el dato 4: 5
Ingresar el dato 5: 4
Ingresar el dato 6: 3
Ingresar el dato 7: 5
```

Archivo: *probabilidad.h*

Función: *probabilidad* Después realiza las operaciones con los datos ingresados y muestra los resultados en pantalla

```

Probabilidad
Ingresa el dato 7: 5
n(U) = 7
      U = {4,3,4,5,4,3,5}

Eventos
A = 4  Aparece 3 veces
B = 3  Aparece 2 veces
C = 5  Aparece 2 veces

Probabilidad de cada evento.
P(A): 0.43
P(B): 0.28
P(C): 0.28

Intersección de los eventos.
AnB: 0.12
AnC: 0.12
BnC: 0.08

Probabilidad de los eventos utilizando la probabilidad condicional.
P(A|B): 0.42
P(A|C): 0.42
P(B|C): 0.28

Probabilidad de los eventos utilizando el teorema de Bayes.
P(B|A): 0.28
P(C|A): 0.28
P(C|B): 0.28

Desea guardar los resultados en un archivo de texto? [S/N]: _

```

Archivo: *Conteo.h*

Función: *MenuConteo*

Muestra un pequeño menú

```

Conteo

-----
CONTEO
1) Combinaciones      3) Regresar
2) Permutaciones      4) Salir
-----
Ingresa una opción (1-4):

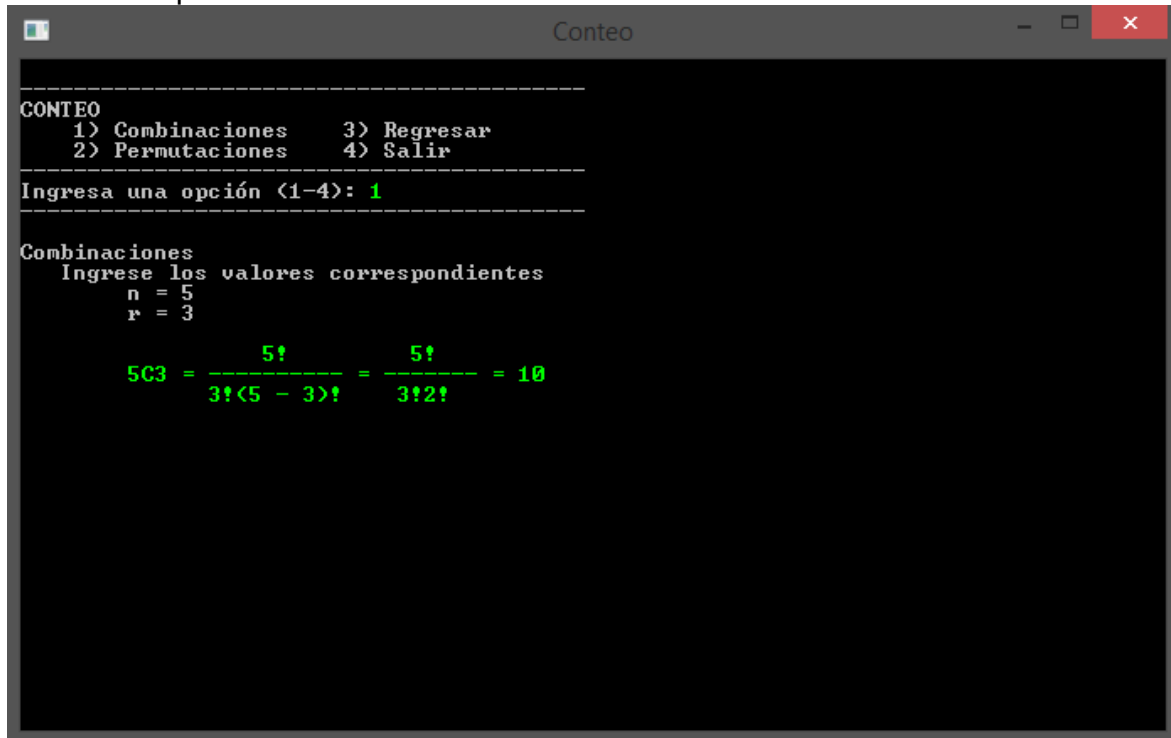
```



Archivo: *Conteo.h*

Función: *combinaciones*

Solicita el valor de n y r, realiza las operaciones necesarias y posteriormente muestra en pantalla el valor del resultado



```
Conteo
-----
CONTEO
1> Combinaciones    3> Regresar
2> Permutaciones    4> Salir
-----
Ingresa una opción <1-4>: 1
-----
Combinaciones
Ingresa los valores correspondientes
n = 5
r = 3

5C3 =  $\frac{5!}{3!(5-3)!} = \frac{5!}{3!2!} = 10$ 
```

Archivo: *Conteo.h*

Función: *permutaciones*

Solicita el valor de n y r, realiza las operaciones necesarias y posteriormente muestra en pantalla el valor del resultado



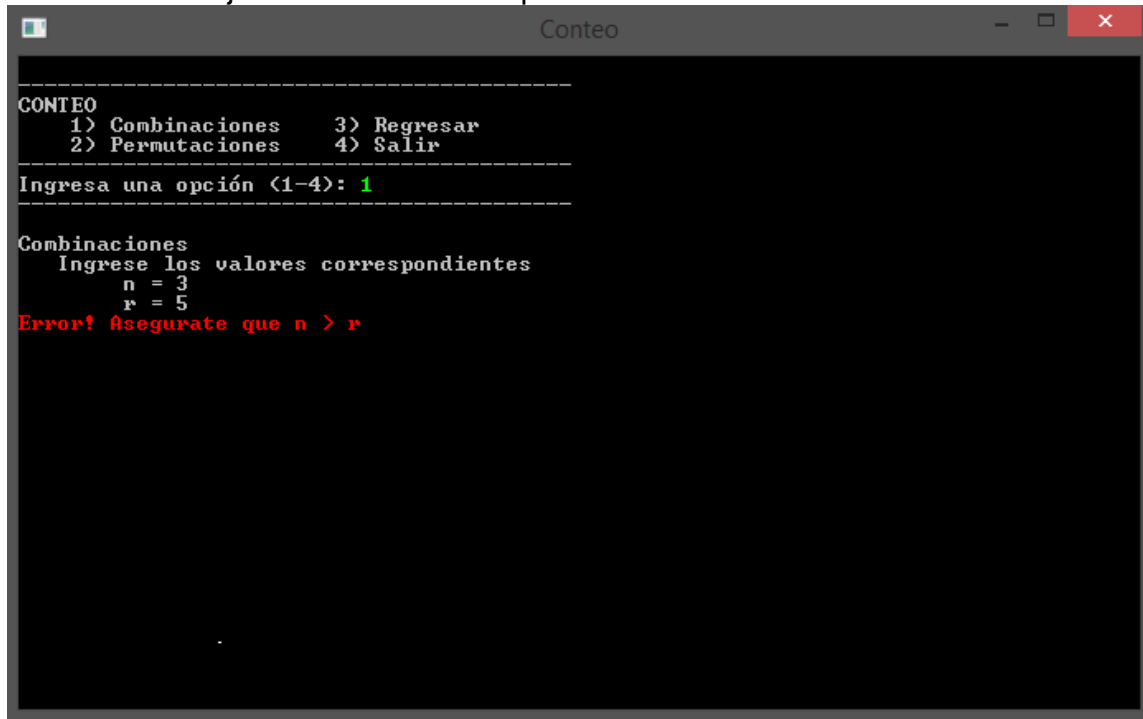
```
Conteo
-----
CONTEO
1> Combinaciones    3> Regresar
2> Permutaciones    4> Salir
-----
Ingresa una opción <1-4>: 2
-----
Permutaciones
Ingresa los valores correspondientes
n = 5
r = 3

5P3 =  $\frac{5!}{(5-3)!} = \frac{5!}{2!} = 60$ 
```

Archivo: *Conteo.h*

Función: *combinaciones y permutaciones*

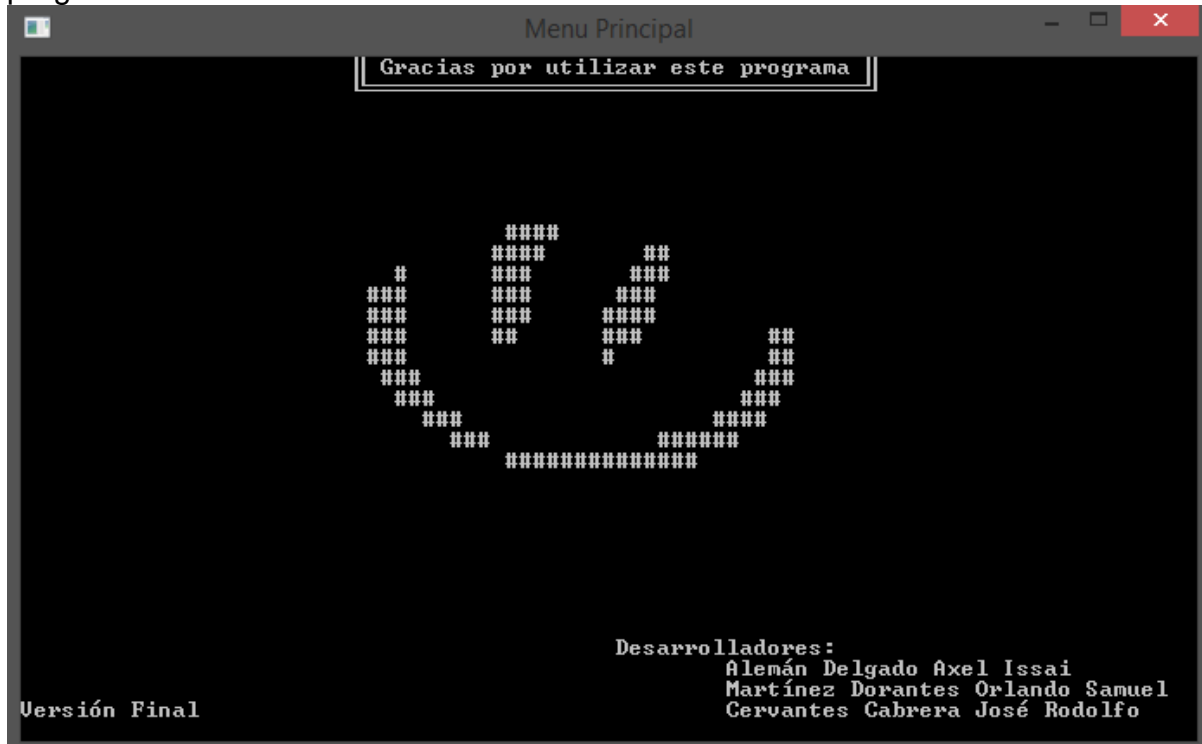
Muestra mensaje de error al no cumplirse la condición  $n > r$



```
Conteo
-----
CONTEO
1> Combinaciones      3> Regresar
2> Permutaciones      4> Salir
-----
Ingresa una opción <1-4>: 1
-----
Combinaciones
Ingresa los valores correspondientes
  n = 3
  r = 5
Error! Asegurate que n > r
```

Archivo: *MenuPrincipal.h* Función: *Salida*

Muestra mensaje de despedida junto con los nombres de los desarrolladores del programa



```
Menu Principal
Gracias por utilizar este programa

#####
#      #####      ##
###    ###      ###
###    ###      ###
###    ###      ###
###    ###      ###
###    ###      ###
###    ###      ###
###    ###      ###
###    ###      ###
#####

Desarrolladores:
Aleján Delgado Axel Issai
Martínez Dorantes Orlando Samuel
Cervantes Cabrera José Rodolfo

Versión Final
```

## Conclusiones.

Hacer el programa fue un gran reto, ya que algunas cosas eran nuevas para nosotros, por lo que tuvimos que investigar, leer y practicar mucho.

La parte más difícil fue a la hora de hacer que el programa extraiga los datos del archivo de texto, lo cual nos tomó varios días lograrlo.

Otra de las partes difíciles del proyecto fue durante la etapa de pruebas, ya que fuimos descubriendo pequeños errores, como por ejemplo, algunas funciones devolvían resultados incorrectos o el programa se cerraba de manera repentina, también tuvimos problemas a la hora de imprimir los resultados ya que en algunas ocasiones no tomaban los decimales, entre otros tantos problemas; concordamos en que esta fue una de las partes difíciles ya que solucionar dichos problemas nos tomó mucho tiempo y en algunas ocasiones era algo frustrante

Dichos problemas los fuimos solucionando poco a poco.

## **Apéndice.**

A continuación se muestra el código fuente de nuestro programa



```

1  #ifndef Estadistica_INCLUDED
2  #define Estadistica_INCLUDED
3
4  #include <string.h>
5  #include "color.h"
6  #include "Grabar_resultados.h"
7
8  //prototipos de funcion
9  float ordenar(float datos[], unsigned int longitud);
10 float media( float datos[], unsigned int longitud);
11 float mediana( float datos[], unsigned int longitud);
12 float moda(float datos[], unsigned int longitud);
13 float varianza(float datos[], unsigned int longitud);
14 float desviacion( float varianza );
15 float cuartiles( int k, float datos[], unsigned int longitud);
16 float percentiles( int k, float datos[], unsigned int longitud);
17
18 //Variables globales
19 float temp = 0; //variable para almacenamiento temporal
20 int i = 0; //contador
21 int j = 0; //contador
22
23 int MenuEstadistica(){
24
25     system("cls");
26     system("mode con: cols=85 lines=33");
27     system("Title Estadistica");
28     Color(NEGRO, BLANCO);
29
30     //-----
31     char Archivo_ubicacion[100];
32     //char * puntero_cadena = Archivo_ubicacion;
33     i = 0;
34
35     //Lee ubicacion del archivo arrastrandolo a la ventana
36     printf("NOTA:\n");
37     printf("• Tanto el archivo como la ubicacion de este no deben contener espacios en el nombre\n");
38     printf("• El archivo debe contener al menos 30 datos\n");
39     printf("• Los datos contenidos en dicho archivo deben estar separados por comas '\n"
40           "  " " Por ejemplo: 1,2,3,4,5,...,n " ");
41     Sleep(3000);
42     system("cls");
43     printf("Arrastra el archivo aqui: ");
44     Color(NEGRO, LVERDE); scanf("%s", Archivo_ubicacion); //lee una línea entera, con espacios incluidos, hasta que pulse intro
45     Color(NEGRO, BLANCO);
46
47     //ALMACENAR LOS DATOS DEL TXT EN UN ARREGLO
48     FILE *myFile; //puntero tipo archivo, para comunicacion entre archivo y programa
49     myFile = fopen(Archivo_ubicacion, "r"); //abre el archivo en modo lectura
50
51     if (myFile == NULL){ //comprobamos que el archivo se haya abierto; si myFile es igual a NULL el archivo no se abrio
52         Color(NEGRO, ROJO); printf("Error leyendo el archivo\n");
53         printf("Asegurate que el archivo cumpla con los requisitos\n");
54         Sleep(3000);
55         free(myFile);
56         MenuEstadistica();
57     }
58
59     //Lee el total de datos en el txt
60     while(!feof(myFile)){
61         fscanf( myFile, "%f,", &temp );
62         i++;
63     }
64
65     if( i < 30){
66         printf("Asegurate que el archivo contenga al menos 30 datos");
67         Sleep(3000);
68         MenuEstadistica();
69     }
70
71     //crea el arreglo con el tamaño necesario
72     float datos[i];
73
74     if(i > 100)
75         system("mode con: cols=85 lines=41");
76
77     //Mueve el cursor al inicio del archivo
78     rewind(myFile);
79
80     printf("\nEl archivo contiene %d elementos\n\n", i);
81
82     //Lee los datos del archivo y los almacena en el arreglo
83     for (j = 0; j < i; j++){
84         fscanf( myFile, "%f,", &datos[j] );
85     }
86
87     printf("Los cuales son:\n");
88     for (j = 0; j < i; j++){
89         Color(NEGRO, MORADO); printf("%.2f ", datos[j]);
90     }
91     printf("\n\n");
92
93     fclose(myFile); //cierra el archivo
94     //-----
95
96     unsigned int longitud = i; //obtiene el tamaño del arreglo
97
98     ordenar(datos, longitud); //ordena arreglo de menor a mayor
99
100    Color(NEGRO, BLANCO); printf("En orden ascendente:\n");
101    for (i = 0; i < longitud; i++){
102        Color(NEGRO, AMARILLO); printf("%.2f ", datos[i]);
103    }
104    Color(NEGRO, BLANCO); printf("\n\n");
105
106    printf("Media = %.2f\n", media( datos, longitud )); //Llama a la funcion mediana e imprime el valor retornado
107
108    printf("Mediana = %.2f\n", mediana( datos, longitud )); //Llama a la funcion mediana e imprime el valor retornado
109
110    printf("Moda = %.2f\n", moda(datos, longitud)); // llama a la funcion moda e imprime el valor retornado
111
112    float varianzal = varianza( datos, longitud ); //Llama a la funcion varianza y almacena el valor retornado en una variable
113    printf("Varianza = %.2f\n", varianzal);
114
115    printf("Desviaci3n est3ndar = %.2f\n", 162, 160, desviacion( varianzal ));
116    // llama a la funcion desviacion estandar e imprime el valor retornado
117
118    printf("Q1 = %.2f\n"
119           "Q2 = %.2f\n"
120           "Q3 = %.2f\n",
121           cuartiles( 1, datos, longitud), cuartiles( 2, datos, longitud), cuartiles( 3, datos, longitud) );
122
123    printf("P30 = %.2f\n"
124           "P90 = %.2f\n\n",
125           percentiles( 30, datos, longitud), percentiles( 90, datos, longitud) );
126
127    //Guardar datos en txt
128    char opcion;
129    printf("¿Desea guardar los resultados en un archivo de texto? [S/N]: ");
130    scanf("%s", &opcion);
131
132    if(opcion == 's' || opcion == 'S'){
133        Guardar_estadistica(Archivo_ubicacion, datos, longitud, varianzal);
134        MenuPrincipal();
135    }
136    else{
137        MenuPrincipal();
138    }
139
140    //funcion para ordenar arreglo de menor a mayor
141    float ordenar(float datos[], unsigned int longitud){
142        temp = 0;
143
144        for(i = 0; i < (longitud - 1); i++){
145            for(j = i + 1; j < longitud; j++){
146                if( datos[j] < datos[i] ){
147                    temp = datos[j];
148                    datos[j] = datos[i];

```

```

149         datos[i] = temp;
150     } //fin if
151 } //fin for anidado
152 } //fin for
153 return Longitud;
154 } //fin funcion
155
156
157 //Funcion para calcular media
158 float media( float datos[], unsigned int longitud){
159
160     float sumatoria = 0;
161
162     for (i = 0; i < longitud; i++){
163         sumatoria += datos[i]; //suma todos los elementos del arreglo
164     }
165     return sumatoria / longitud; //divide la suma entre el total de elementos obteniendo asi la media
166 } //Fin funcion
167
168 //Funcion calcular mediana
169 float mediana( float datos[], unsigned int longitud){
170
171     if (longitud % 2 == 0){ //si es par...
172         return (datos[(longitud/2) - 1] + datos[(longitud/2)] / 2.00 ); //...suma los dos valores centrales y obtiene su promedio
173     }
174     else{ //si es impar...
175         return datos[(longitud/2)]; //...Devuelve el valor central
176     }
177 } //fin funcion
178
179 //Funcion calcular varianza (Xn-media)^2 / n
180 float varianza(float datos[], unsigned int longitud){
181
182     float medial = media( datos, longitud );
183     float sumatoria = 0;
184
185     for (i = 0; i < longitud; i++){
186         sumatoria += pow( (datos[i] - medial), 2); //Sumatoria (x - media)^2
187     }
188     return sumatoria / longitud; //divide la sumatoria entre el total de elementos obteniendo asi la media
189 } //Fin funcion
190
191 //funcion calcular desviacion estandar
192 float desviacion( float varianza ){
193     return sqrt(varianza);
194 } //fin funcion
195
196 //Funcion calcular cuartiles Qk = (k * n) / 4
197 float cuartiles( int k, float datos[], unsigned int longitud){
198
199     float Q = 0;
200
201     //determina que formula utilizar
202     if( (longitud%2 == 0){ //si el total de datos es par...
203         Q = (k * longitud) / 4.00; //..utiliza esta formula para obtener posicion del cuartil solicitado
204     }
205     else //sino...
206         Q = (k * (longitud+1)) / 4.00; //...utiliza esta formula para obtener posicion del cuartil solicitado
207
208     //devuelve el resultado
209     if ( (k * longitud) % 4 == 0 ){
210         if((longitud % 2 == 0){
211             return ( datos[(int)Q] + datos[(int)Q - 1]) / 2;
212         }
213         else{
214             return datos[(int)Q - 1];
215         }
216     }
217     else{ //interpolamos Q = d2(d3) + d1
218         float d1 = datos[(int)Q - 1]; //Valor que se encuentra en la posicion de la parte entera
219         float d2 = Q - (int)Q; //Parte decimal
220         float d3 = datos[(int)Q] - datos[(int)Q - 1]; // (Q + 1) - Q
221         return d2*(d3) + d1;
222     }
223 } //fin funcion
224
225 //Funcion calcular percentiles Pk = (k * n) / 100
226 float percentiles( int k, float datos[], unsigned int longitud){
227
228     float P = 0;
229
230     if((longitud%2 == 0){ //si el total de datos es par...
231         P = (k * longitud) / 100.00; //..utiliza esta formula para obtener posicion del percentil solicitado
232     }
233     else //sino...
234         P = (k * (longitud+1)) / 100.00; //...utiliza esta formula para obtener posicion del percentil solicitado
235
236
237     if ( (k * longitud) % 100 == 0 ){
238         return datos[(int)P - 1];
239     }
240     else{
241         float d1 = datos[(int)P - 1];
242         float d2 = P - (int)P;
243         float d3 = datos[(int)P] - datos[(int)P - 1];
244         return d2*(d3) + d1;
245     }
246 } //fin funcion
247
248 //Funcion para calcular moda
249 float moda(float datos[], unsigned int longitud){
250
251     float count=0;
252     float max_conteo=0;
253     float max_variable=0;
254
255     for(i=0;i<longitud;++i){
256
257         i=0;
258         while(i<longitud) {
259             j=0;
260             count=0;
261             while(j<longitud) {
262                 if(datos[i]==datos[j]) {
263                     count++;
264                 }
265                 if (count>max_conteo) {
266                     max_conteo=count;
267                     max_variable=datos[i];
268                 }
269                 j++;
270             }
271             i++;
272         }
273     }
274     return max_variable;
275 } //fin funcion
276
277 #endif

```



```
1  #ifndef probabilidad_INCLUDED
2  #define probabilidad_INCLUDED
3
4  #include "color.h"
5  #include "Grabar_resultados.h"
6
7  float probabilidad(){
8
9      system("cls");
10     system("mode con: cols=85 lines=33");
11     system("Title Probabilidad");
12
13     //definición de las variables
14     char opcion;
15     int longitud = 0, i = 0, A = 0, B = 0, C = 0;
16     float probA, probB, probC;
17     float interAB, interAC, interBC;
18     float condAB, condAC, condBC;
19     float bayesBA, bayesCA, bayesCB;
20
21     printf("Ingresa el tama%co que tendr%c el espacio muestra (-1 para salir): ", 164, 160);
22     scanf("%d", &longitud);
23     if(longitud == -1)
24         MenuPrincipal();
25
26     int Espacio[longitud];
27
28     for(i = 0; i < longitud; i++){
29         Color(NEGRO, BLANCO);   printf("Ingresa el dato %d: ", (i+1));
30         Color(NEGRO, LVERDE);   scanf("%d", &Espacio[i]);
31     }
32
33     //Imprime el espacio muestral
34     Color(NEGRO, BLANCO);   printf("\n\tU = {");
35     for(i = 0; i < longitud; i++){
36         Color(NEGRO, LVERDE);   printf("%d,", Espacio[i]);
37     }
38     Color(NEGRO, BLANCO);   printf("}\n\n");
39
40     //Obtiene el valor de A, B y C, y sus repeticiones
41     int repA = 0, repB = 0, repC = 0;
42     A = Espacio[0];
43
44     for ( i = 0; i < longitud; i++ ){
45         if(Espacio[i] != A){
46             if(Espacio[i] != C){
47                 if(B == 0){
48                     B = Espacio[i];
49                 }else{
50                     C = Espacio[i];
51                 }
52             }
53         }
54     }
55
56     for ( i = 0; i < longitud; i++ ){
57         if(Espacio[i] == A)
58             repA++;
59         if(Espacio[i] == B)
60             repB++;
61         if(Espacio[i] == C)
62             repC++;
63     }
64     Color(NEGRO, BLANCO);
65     printf("\nA = %d\n", A);
66     printf("B = %d\n", B);
67     printf("C = %d\n", C);
68
69     //Operaciones de las probabilidades
70     probA = trunc(( (float)repA / longitud ) * 1000.0) / 1000.0;
71     probB = trunc(( (float)repB / longitud ) * 1000.0) / 1000.0;
72     probC = trunc(( (float)repC / longitud ) * 1000.0) / 1000.0;
73     interAB = trunc((probA * probB) * 1000.0) / 1000.0;
74     interAC = trunc((probA * probC) * 1000.0) / 1000.0;
75     interBC = trunc((probB * probC) * 1000.0) / 1000.0;
76     condAB = trunc((interAB / probB) * 1000.0) / 1000.0;
77     condAC = trunc((interAC / probC) * 1000.0) / 1000.0;
78     condBC = trunc((interBC / probC) * 1000.0) / 1000.0;
79     bayesBA = trunc(((condAB * probB) / probA) * 1000.0) / 1000.0;
80     bayesCA = trunc(((condAC * probC) / probA) * 1000.0) / 1000.0;
81     bayesCB = trunc(((condBC * probC) / probB) * 1000.0) / 1000.0;
82
83     Color(NEGRO, BLANCO);   //regresa los colores de la ventana a los valores por defecto
84
85     //Probabilidades individuales
86     printf("\nProbabilidad de cada evento.\n");
87     printf(" P(A): %.2f\n", probA);
88     printf(" P(B): %.2f\n", probB);
89     printf(" P(C): %.2f\n\n", probC);
90
91     //Intersecciones
92     printf("Intersecci%cn de los eventos.\n", 162);
93     printf(" AnB: %.2f\n", interAB);
94     printf(" AnC: %.2f\n", interAC);
95     printf(" BnC: %.2f\n\n", interBC);
96
97     //Probabilidad Condicional
98     printf("Probabilidad de los eventos utilizando la probabilidad condicional.\n");
99     printf(" P(A|B): %.2f\n", condAB);
100    printf(" P(A|C): %.2f\n", condAC);
101    printf(" P(B|C): %.2f\n\n", condBC);
102
103    //Teorema de Bayes
104    printf("Probabilidad de los eventos utilizando el teorema de Bayes.\n");
105    printf(" P(B|A): %.2f\n", bayesBA);
106    printf(" P(C|A): %.2f\n", bayesCA);
107    printf(" P(C|B): %.2f\n\n", bayesCB);
108
109    //Guardar datos en txt
110    printf("%cDesea guardar los resultados en un archivo de texto? [S/N]: ");
111    scanf("%s", &opcion);
112
113    if(opcion == 'S' || opcion == 's'){
114        Guardar_probabilidad(Espacio, longitud, A, B, C, probA, probB, probC, interAB, interAC, interBC, condAB, condAC, condBC, bayesBA,
115        bayesCA, bayesCB);
116        probabilidad();
117    }
118    else
119        probabilidad();
120
121    #endif
```



```

#ifndef Conteo_INCLUDED
#define Conteo_INCLUDED

#include "MenuPrincipal.h"
#include "color.h"

//Prototipos de funcion
unsigned long long int factorial ( int num );
unsigned long long int combinaciones ();
unsigned long long int permutaciones ();

//Variables globales
unsigned int n, r, x;
unsigned long long int n_fact, r_fact, x_fact;
unsigned long long int combinacionesRes, permutacionesRes;

int MenuConteo(){

    system("cls");
    system("mode con: cols=85 lines=33");
    system("Title Conteo");
    Color(NEGRO, BLANCO);

    int opcion;

    printf("\n-----\n");
    printf("CONTEO\n");
    printf("%20s %15s", "1) Combinaciones", "3) Regresar\n");
    printf("%20s %12s", "2) Permutaciones", "4) Salir\n");
    printf("-----\n");
    printf("Ingresa una opci%c\n (1-4): ", 162);
    Color(NEGRO, LVERDE);    scanf("%d", &opcion);
    Color(NEGRO, BLANCO);    printf("-----\n\n");

    switch(opcion){

        case 1:
            combinacionesRes = combinaciones();
            printf("\n\t\t %u%c \t\t %u%c", n, 33, n, 33);
            printf("\n\t%uC%u = ----- = ----- = %llu", n, r, combinacionesRes);
            printf("\n\t\t %u%c(%u - %u)%c\t\t %u%c%u%c\n", r, 33, n, r, 33, r, 33, (n-r), 33);
            Sleep(3000);
            MenuConteo();
            break;

        case 2:
            permutacionesRes = permutaciones ();
            printf("\n\t\t %u%c \t\t %u%c", n, 33, n, 33);
            printf("\n\t%uP%u = ----- = ----- = %llu", n, r, permutacionesRes);
            printf("\n\t\t (%u - %u)%c \t\t %u%c\n", n, r, 33, (n-r), 33);
            Sleep(3000);
            MenuConteo();
            break;
    }
}

```

```

        case 3:
            MenuPrincipal();
            break;

        case 4:
            Salida();
            break;

        default:
            Color(NEGRO, ROJO);    printf("\tOPCI%cN INVALIDA!! \n    Por favor intenta de nuevo", 224);
            Sleep(2000);
            MenuConteo();
            break;
    }
}

//Funcion para factoriales necesarios
unsigned long long int factorial(int num){

    unsigned long long int fact = 1, i;

    for ( i = 1; i <= num; i++ ){
        fact = fact * i;
    }
    return fact;
}

//Funcion para combinaciones    C = n! / r!(n - r)!
unsigned long long int combinaciones (){

    printf("Combinaciones\n");
    printf("    Ingrese los valores correspondientes\n");
    printf("\tn = ");
    scanf("%u", &n);
    printf("\tr = ");
    scanf("%u", &r);

    if(n > r){ //Valida que n > r...
        //Llama a la funcion factorial para obtener el factorial de cada valor
        n_fact = factorial( n );
        r_fact = factorial( r );
        x_fact = factorial( n - r );

        //retorna el total de combinaciones
        Color(NEGRO, LVERDE);    return n_fact / (r_fact * x_fact);
    }
    else{ //... en caso contrario imprime un mensaje de error
        Color(NEGRO, ROJO);    printf("Error! Asegurate que n > r\n");
    }
    Sleep(2000);
    MenuConteo();
}

//Funcion para permutaciones    P = n! / (n - r)!
unsigned long long int permutaciones (){

```

```
printf("Permutaciones\n");

printf("  Ingrese los valores correspondientes\n");
printf("\tn = ");
scanf("%u", &n);
printf("\tr = ");
scanf("%u", &r);
if(n > r){
```

```
n_fact = factorial( n );
x_fact = factorial( n - r );

    Color(NEGRO, LVERDE);    return n_fact / x_fact;
}
else{
    Color(NEGRO, ROJO);    printf("Error! Asegurate que n > r\n");
}
Sleep(2000);
MenuConteo();
}

#endif
```

```

1  #ifndef Grabar_resultados_INCLUDED
2  #define Grabar_resultados_INCLUDED
3
4  #include <stdio.h>
5  #include <windows.h>
6  #include <sys/stat.h>
7  #include <time.h>
8  #include <math.h>
9
10 float ordenar2(float datos[], unsigned int Longitud);
11 float media2( float datos[], unsigned int Longitud);
12 float mediana2( float datos[], unsigned int Longitud);
13 float moda2(float datos[], unsigned int Longitud);
14 float desviacion2( float varianza1 );
15 float cuartiles2( int k, float datos[], unsigned int Longitud);
16 float percentiles2( int k, float datos[], unsigned int Longitud);
17
18 int count1 = 0;
19 int count2 = 0;
20
21 float Guardar_estadistica(char Archivo_ubicacion, float datos[], unsigned int Longitud, float varianza1){
22
23     float Medial = media2( datos, Longitud );
24     float Mediana1 = mediana2( datos, Longitud );
25     float Moda1 = moda2(datos, Longitud);
26     float desviacion1 = desviacion2( varianza1 );
27     float Q1 = cuartiles2( 1, datos, Longitud);
28     float Q2 = cuartiles2( 2, datos, Longitud);
29     float Q3 = cuartiles2( 3, datos, Longitud);
30     float P30 = percentiles2( 30, datos, Longitud);
31     float P90 = percentiles2( 90, datos, Longitud);
32
33     mkdir("Resultados"); //Crea carpeta
34
35     //definir nombre de archivo con fecha y hora
36     time_t tiempo = time(0);
37     struct tm *tlocal = localtime(&tiempo);
38     char Nombre_Archivo[128];
39     strftime(Nombre_Archivo,128,"Resultados/Estadistica_%d-%m-%y_%H-%M-%S.txt",tlocal);
40
41     //crea el archivo de texto y lo abre en modo escritura
42     FILE *fptr = fopen(Nombre_Archivo, "w");
43     if (fptr == NULL)
44     {
45         printf("No se pudo abrir el archivo");
46         return 0;
47     }
48     fprintf(fptr, "\nEl archivo contiene %u elementos\n\n", Longitud);
49     fprintf(fptr, "Los cuales son:\n");
50     for (count1 = 0; count1 < Longitud; count1++){
51         fprintf(fptr, "%.2f ", datos[count1]);
52     }
53     fprintf(fptr, "\n\nMedia = %.2f\n", Medial);
54     fprintf( fptr, "Mediana = %.2f\n", Mediana1 );
55     fprintf(fptr, "Moda = %.2f\n", Moda1);
56     fprintf(fptr, "Varianza = %.2f\n", varianza1);
57     fprintf(fptr, "Desviación estándar = %.2f\n", desviacion1);
58     fprintf( fptr, "Q1 = %.2f\nQ2 = %.2f\nQ3 = %.2f\n", Q1, Q2, Q3);
59     fprintf( fptr, "P30 = %.2f\nP90 = %.2f\n\n", P30, P90);
60
61     fclose(fptr);
62     printf("\nLos resultados fueron guardados en %s\n", Nombre_Archivo);
63     Sleep(2000);
64     return 0;
65 }
66
67 float media2( float datos[], unsigned int Longitud){
68
69     float sumatoria = 0;
70
71     for (count1 = 0; count1 < Longitud; count1++){
72         sumatoria += datos[count1]; //suma todos los elementos del arreglo
73     }
74     return sumatoria / Longitud; //divide la suma entre el total de elementos obteniendo asi la media
75 } //Fin funcion
76
77 float mediana2( float datos[], unsigned int Longitud){
78
79     if (Longitud % 2 == 0){ //si es par...
80         return( (datos[(Longitud/2) - 1] + datos[Longitud/2]) / 2.00 ); //...suma los dos valores centrales y obtiene su promedio
81     }
82     else{ //si es impar...
83         return datos[Longitud/2]; //...Devuelve el valor central
84     }
85 } //fin funcion
86
87 float desviacion2( float varianza ){
88     return sqrt(varianza);
89 } //fin funcion
90
91 float cuartiles2( int k, float datos[], unsigned int Longitud){
92
93     float Q = 0;
94
95     //determina que formula utilizar
96     if( Longitud%2 == 0){ //si el total de datos es par...
97         Q = (k * Longitud) / 4.00; //...utiliza esta formula para obtener posicion del cuartil solicitado
98     }
99     else //sino...
100         Q = ( k * (Longitud+1) ) / 4.00; //...utiliza esta formula para obtener posicion del cuartil solicitado
101
102     //devuelve el resultado
103     if ( (k * Longitud) % 4 == 0 ){
104         if(Longitud % 2 == 0){
105             return ( datos[(int)Q] + datos[(int)Q - 1]) / 2;
106         }
107         else{
108             return datos[(int)Q - 1];
109         }
110     }
111     else{ //interpolamos Q = d2(d3) + d1
112         float d1 = datos[(int)Q - 1]; //Valor que se encuentra en la posición de la parte entera
113         float d2 = Q - (int)Q; //Parte decimal
114         float d3 = datos[(int)Q] - datos[(int)Q - 1]; // (Q + 1) - Q
115         return d2*(d3) + d1;
116     }
117 } //fin funcion
118

```

```

119 float percentiles2( int k, float datos[], unsigned int Longitud){
120
121     float P = 0;
122
123     if(Longitud%2 == 0){ //si el total de datos es par...
124         P = (k * Longitud) / 100.00; //..utiliza esta formula para obtener posicion del percentil solicitado
125     }
126     else //sino...
127         P = ( k * (Longitud+1) ) / 100.00; //...utiliza esta formula para obtener posicion del percentil solicitado
128
129     if ( (k * Longitud) % 100 == 0 ){
130         return datos[(int)P - 1];
131     }
132     else{
133         float d1 = datos[(int)P - 1];
134         float d2 = P - (int)P;
135         float d3 = datos[(int)P] - datos[(int)P - 1];
136         return d2*(d3) + d1;
137     }
138 } //fin funcion
139
140 float moda2(float datos[], unsigned int Longitud){
141
142     float count=0;
143     float max_conteo=0;
144     float max_variable=0;
145
146     for(count1=0;count1<Longitud;++count1){
147
148         count1=0;
149         while(count1<Longitud) {
150             count2=0;
151             count=0;
152             while(count2<Longitud) {
153                 if(datos[count1]==datos[count2]) {
154                     count++;
155                 }
156                 if (count>max_conteo) {
157                     max_conteo=count;
158                     max_variable=datos[count1];
159                 }
160                 count2++;
161             }
162             count1++;
163         }
164     }
165     return max_variable;
166 } //fin funcion
167
168 //-----
169
170 float Guardar_probabilidad(int A, int B, int C, float probA, float probB, float probC,
171                             float interAB, float interAC, float interBC,
172                             float condAB, float condAC, float condBC, float bayesBA,
173                             float bayesCA, float bayesCB){
174
175     //definir nombre de archivo con fecha y hora
176     time t tiempo = time(0);
177     struct tm *tlocal = localtime(&tiempo);
178     char Nombre_Archivo[128];
179     strftime(Nombre_Archivo,128,"Probabilidad_%d-%m-%y_%H-%M-%S.txt",tlocal);
180
181     //crea el archivo de texto y lo abre en modo escritura
182     FILE *fptr = fopen(Nombre_Archivo, "w");
183     if (fptr == NULL)
184     {
185         printf("No se pudo abrir el archivo");
186         return 0;
187     }
188
189     fprintf(fptr, "Valor de la probabilidad de A: %d\n", A);
190     fprintf(fptr, "Valor de la probabilidad de B: %d\n", B);
191     fprintf(fptr, "Valor de la probabilidad de C: %d\n\n", C);
192
193     fprintf(fptr, "Probabilidad de cada evento.\n");
194     fprintf(fptr, " P(A): %.2f\n", probA);
195     fprintf(fptr, " P(B): %.2f\n", probB);
196     fprintf(fptr, " P(C): %.2f\n\n", probC);
197
198     fprintf(fptr, "Intersección de los eventos.\n");
199     fprintf(fptr, " AnB: %.2f\n", interAB);
200     fprintf(fptr, " AnC: %.2f\n", interAC);
201     fprintf(fptr, " BnC: %.2f\n\n", interBC);
202
203     fprintf(fptr, "Probabilidad de los eventos utilizando la probabilidad condicional.\n");
204     fprintf(fptr, " P(A|B): %.2f\n", condAB);
205     fprintf(fptr, " P(A|C): %.2f\n", condAC);
206     fprintf(fptr, " P(B|C): %.2f\n\n", condBC);
207
208     fprintf(fptr, "Probabilidad de los eventos utilizando el teorema de Bayes.\n");
209     fprintf(fptr, " P(B|A): %.2f\n", bayesBA);
210     fprintf(fptr, " P(C|A): %.2f\n", bayesCA);
211     fprintf(fptr, " P(C|B): %.2f\n\n", bayesCB);
212
213     fclose(fptr);
214
215     printf("\nLos resultados fueron guardados en %s\n", Nombre_Archivo);
216     Sleep(2000);
217     return 0;
218 }
219
220 #endif

```



## Color.h

```
1  #ifndef Color_INCLUDED
2  #define Color_INCLUDED
3
4  #include <windows.h>
5  //Definicion de los colores
6  #define NEGRO 0
7  #define OSCURO 128
8  #define LVERDE 10
9  #define BLANCO 7
10 #define ROJO 12
11 #define AMARILLO 14
12 #define MORADO 13
13 #define BLANCO_ROJO 252
14
15 void Color(int Background, int Text){ // Función para cambiar el color del fondo y/o pantalla
16
17     HANDLE Console = GetStdHandle(STD_OUTPUT_HANDLE); // Tomamos la consola.
18
19     // Para cambiar el color, se utilizan números desde el 0 hasta el 255.
20     // Pero, para convertir los colores a un valor adecuado, se realiza el siguiente cálculo.
21     int New_Color= Text + (Background * 16);
22
23     SetConsoleTextAttribute(Console, New_Color); // Guardamos los cambios en la Consola.
24 }
25
26 #endif
```