Examen

Integrantes:
Mauro Bernal

Taller de Desarrollo de Aplicaciones

25/03/2021

Objetivo

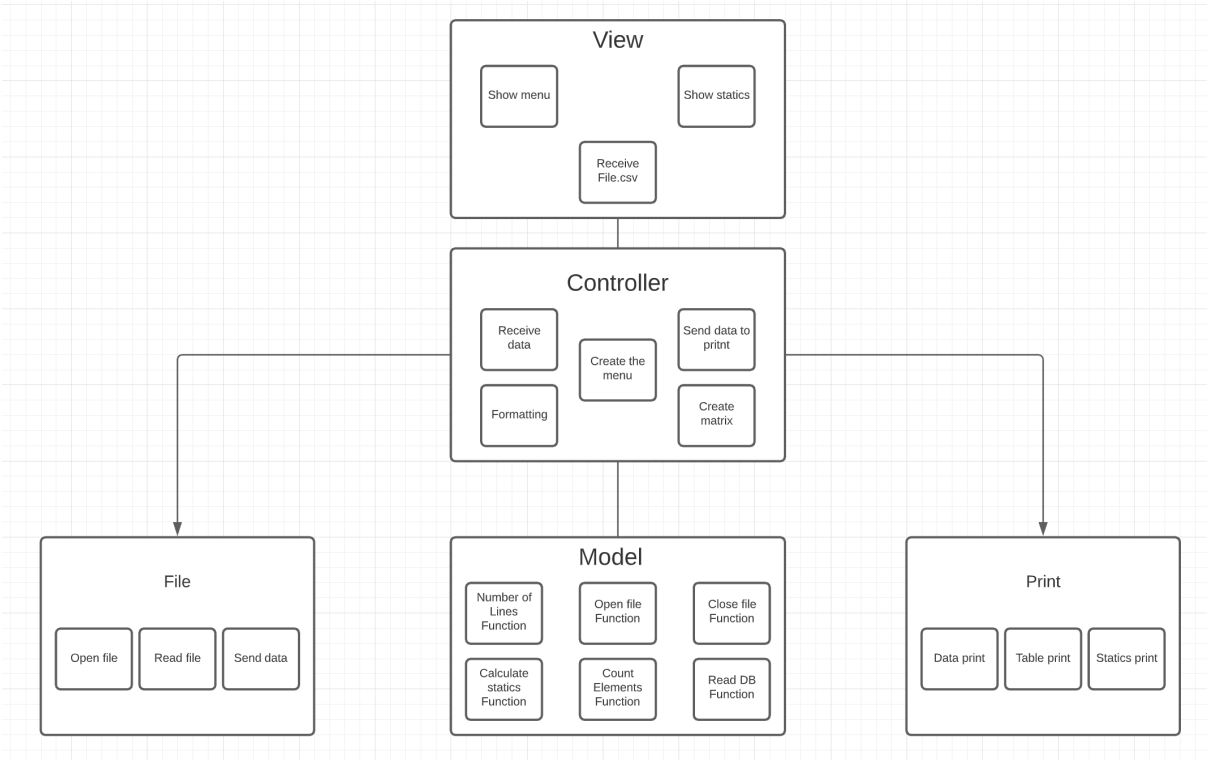
El objetivo de este examen es desarrollar un código que pueda ir leyendo los datos de un archivo csv, los almacenaremos gracias a la técnica de *double pointer arithmetic*. Una vez almacenado, calcularemos la media, mediana, moda, desviación estándar, rangos y distancia intercuartil. Este programa lo realizaremos de manera modular, gracias al modelo MVC (Modelo, Vista, Controlador) así como lo vimos en clase.

Manual de Usuario

Cuando se inicia el programa, este leerá el archivo llamado “data.csv” en automático, te desplegará un menú diciéndote si quieres imprimir los datos leídos (y almacenados) junto con la media, mediana, moda, desviación estándar, rangos y distancia intercuartil o decidas acabar el programa, solo tiene estas dos opciones. Dentro de la carpeta zip, viene un archivo llamado “data.csv” el cual personalmente cree para la realización de este examen.

Para compilar se usa el comando “make” y se ejecuta con “./eje”.

MVC



Pseudocódigo

//
// prac1tda.c
// Created by Mauro Bernal on 25/03/21.
//

Vista

```
function menuOne(char **fileName)
{
	system("clear");
	imprime("Name of File: ");
	*fileName = "data.csv";
	//lee("%m[^\n]", fileName); //Reads until it finds a \n
	imprime("Reading from: %s...\n", *fileName);
}
fin funcion

function menuTwo(function)
{
	int option;

	system("clear");
	imprime("\nWhat do you wanna do?:\n[1] Print data. \n[2] Exit.\nAnswer: ");
	lee("%d", &option);
	imprime("\n");
	system("clear");
	regresa(option);
}
fin funcion
```

Controlador

```
funcion main (void)
{
	char* fileName=NULL;
	int numLines = 0, numElements;
	int option;

	//Calling function to get name
	menuOne(&fileName);

	numLines = NumberLines(fileName); //Calling function to count how many lines
	numElements = countElements(fileName); //Calling function to count elements in line

	float **Matrix = malloc(sizeof(float) * numLines * numElements);

	readDB(Matrix, fileName, numElements);

	do{
	option = menuTwo();
	if(option == 1)
	{
	printMatrix(Matrix, &numLines, &numElements);
	lee("\nPress [enter] to go back to the menu.\n");
	fpurge(stdin);
	getchar();
	}
	}hasta(option != 2);


	libera(Matrix);
	regresa 0;
}

fin funcion
```

Modelo

```
#include "DataShell.h"
#include <math.h>

//Function that prints a Matrix
funcion printMatrix (float **Matrix, int *numLines, int *numElements)
{
        int mitad;
        float prom, x=0, desv;

        //CALCULAR MEDIANA
        repite(int i = 0; i < *numLines; i++)
        {
        repite(int j = 0; j < *numElements; j ++)
        {
        imprime("%.6f ", *((*(Matrix+i))+j));
        prom += *((*(Matrix+i))+1);
        x++;
        }
        fin repite
        imprime("\n");
   }
        fin repite


        //CALCULAR DESVIACION ESTANDAR
        repite(int i = 0; i < *numLines; i++)
        {
        repite(int j = 0; j < *numElements; j ++)
        {
        //imprime("%.6f ", *((*(Matrix+i))+j));
        desv += pow( *((*(Matrix+i))+1) - prom,2);
        desv = sqrt(desv);
        } fin repite
        imprime("\n");
        }
        fin repite


        prom = prom/x;
        desv = (desv/x)/2;
        imprime("\n%lf\n", *((*(Matrix+2))+1));
        imprime("\nMedia = %f\n",prom);
        imprime("\nDesviación estándar = %f\n",desv);
```

```
}

fin funcion

//Function that reads DB and stores it in a Matrix
funcion readDB (float **Matrix, char *fileName, int numElements)
{
	char buff[255];
	char * value;
	float *ptr = NULL;
	int cont = 0, n =0;

	FILE *file = openMyfile(fileName);

	mientras(!feof(file))
	{
	flee(file, "%s", buff);

	//numElements = countElements(buff);
	float *ptr = malloc(sizeof(float) * numElements);
	value = strtok(buff, ",");

	mientras(value != NULL)
	{
	ptr[cont] = atof(value); //Stores it in ptr GUARDO LOS ELEMENTOS EN EL
POINTER
	//imprime("%.6f ", ptr[cont]);
	cont ++;
	value = strtok(NULL, ",");
	}
	//imprime("\n");
	cont = 0;
	Matrix[n] = ptr; //MATRIX AHORA APUNTA A PTR
	n ++;

	}
	closeMyfile(file);


	regresa 0;
}
fin funcion

//Function that tells how many elements does a csv line has
int countElements(char *fileName)
{
	int numElements = 0;
	char buff[255];
```

```
	FILE *file = openMyfile(fileName);
	flee(file, "%s", buff);
	char* value = strtok(buff, ",");

	mientras(value != NULL)
	{
	numElements ++;
	value = strtok(NULL, ",");
	}
	closeMyfile(file);
	regresa numElements;
}

//Function that opens a file
FILE* openMyfile (char *fileName)
{
	FILE *file = fopen(fileName, "r");

	si(file == NULL)
	{
	imprime("Couln´t open file\n");
	exit(0);
	}
	regresa file;
}
fin funcion

//Function that closes a file
funcion closeMyfile(FILE *file)
{
	fclose(file);
}
fin funcion

//Function that return number of lines
funcion NumberLines(char *fileName)
{
	int numLines = 0;
	char buff[255];

	FILE *file = openMyfile(fileName);

	mientras(!feof(file))
	{
	flee(file, "%s", buff);
	numLines ++;
	}
```

```
        closeMyfile(file);

        regresa numLines;
}
```

fin funcion

Código

Controlador

```
#include "DataShell.h"

int main (void)
{
   char* fileName=NULL;
   int numLines = 0, numElements;
   int option;

   //Calling function to get name
   menuOne(&fileName);

   numLines = NumberLines(fileName); //Calling function to count how many lines
   numElements = countElements(fileName); //Calling function to count elements in line

   float **Matrix = malloc(sizeof(float) * numLines * numElements);

   readDB(Matrix, fileName, numElements);

   do{
      option = menuTwo();
      if(option == 1)
      {
         printMatrix(Matrix, &numLines, &numElements);
         printf("\nPress [enter] to go back to the menu.\n");
         fpurge(stdin);
         getchar();
      }
   }while(option != 2);


   free(Matrix);
   return 0;
}
```

Vista

```
#include "DataShell.h"

void menuOne(char **fileName)
{
   system("clear");
   printf("Name of File: ");
   *fileName = "data.csv";
   //scanf("%m[^\n]", fileName); //Reads until it finds a \n
   printf("Reading from: %s...\n", *fileName);
}

int menuTwo(void)
{
   int option;

   system("clear");
   printf("\nWhat do you wanna do?:\n[1] Print data. \n[2] Exit.\nAnswer: ");
   scanf("%d", &option);
   printf("\n");
   system("clear");
   return(option);
}
```

Modelo

```
#include "DataShell.h"
#include <math.h>

//Function that prints a Matrix
void printMatrix (float **Matrix, int *numLines, int *numElements)
{
   int mitad;
   float prom, x=0, desv;

   //CALCULAR MEDIANA
   for(int i = 0; i < *numLines; i++)
   {
      for(int j = 0; j < *numElements; j ++)
      {
         printf("%.6f ", *((*(Matrix+i))+j));
         prom += *((*(Matrix+i))+1);
         x++;
      }
      printf("\n");
   }


   //CALCULAR DESVIACION ESTANDAR
   for(int i = 0; i < *numLines; i++)
   {
      for(int j = 0; j < *numElements; j ++)
```

```
      {
         //printf("%.6f ", *((*(Matrix+i))+j));
         desv += pow( *((*(Matrix+i))+1) - prom,2);
         desv = sqrt(desv);
      }
      printf("\n");
   }


   prom = prom/x;
   desv = (desv/x)/2;
   printf("\n%lf\n", *((*(Matrix+2))+1));
   printf("\nMedia = %f\n",prom);
   printf("\nDesviación estándar = %f\n",desv);


}

//Function that reads DB and stores it in a Matrix
void readDB (float **Matrix, char *fileName, int numElements)
{
   char buff[255];
   char * value;
   float *ptr = NULL;
   int cont = 0, n =0;

   FILE *file = openMyfile(fileName);

   while(!feof(file))
   {
      fscanf(file, "%s", buff);

      //numElements = countElements(buff);
      float *ptr = malloc(sizeof(float) * numElements);
      value = strtok(buff, ",");

      while(value != NULL)
      {
         ptr[cont] = atof(value); //Stores it in ptr GUARDO LOS ELEMENTOS EN EL POINTER
         //printf("%.6f ", ptr[cont]);
         cont ++;
         value = strtok(NULL, ",");
      }
      //printf("\n");
      cont = 0;
      Matrix[n] = ptr; //MATRIX AHORA APUNTA A PTR
      n ++;

   }
   closeMyfile(file);

   //printf("\n%.6f %.6f\n", *((*(Matrix+49))+0), *((*(Matrix+49))+1) );
```

```
    return;
}

//Function that tells how many elements does a csv line has
int countElements(char *fileName)
{
    int numElements = 0;
    char buff[255];

    FILE *file = openMyfile(fileName);
    fscanf(file, "%s", buff);
    char* value = strtok(buff, ",");

    while(value != NULL)
    {
        numElements ++;
        value = strtok(NULL, ",");
    }
    //printf("num = %d ", numElements);
    closeMyfile(file);
    return numElements;
}

//Function that opens a file
FILE* openMyfile (char *fileName)
{
    FILE *file = fopen(fileName, "r");

    if(file == NULL)
    {
        printf("Couln´t open file\n");
        exit(0);
    }
    return file;
}

//Function that closes a file
void closeMyfile(FILE *file)
{
    fclose(file);
}

//Function that return number of lines
int NumberLines(char *fileName)
{
    int numLines = 0;
    char buff[255];

    FILE *file = openMyfile(fileName);

    while(!feof(file))
```

```
    {
        fscanf(file, "%s", buff);
        numLines ++;
    }

    closeMyfile(file);

    return numLines;
}
```

Conclusión

En este examen logré poner en práctica todos los conocimientos que he obtenido hasta la fecha en la materia, manipulé los datos que recopile del archivo csv para poder realizar unos cálculos estadísticos, algo que se aplica hoy en día en Inteligencia Artificial. Además de eso, observe la importancia del modelo MVC, pues facilita bastante al momento de programar, pues te ayuda a entender de mejor manera cómo vas a hacer lo que se te está pidiendo y aterrizas mejor las ideas.