	Carátula para entrega de prácticas
Facultad de Ingeniería	Laboratorio de docencia

Profesor: MARCO ANTONIO MARTINEZ QUINTANA

Asignatura: ESTRUCTURA DE DATOS Y ALGORITMOS I

Grupo: 17

Proyecto: BASE DE DATOS

Alumno: José Luis Arroyo Chavarría

No. de Lista : 5

Semestre: 2

Fecha de entrega:

Observaciones:

CALIFICACIÓN: _____

Objetivo:

Tener una base de guardado de una gran variedad de documentos y con su clasificación de estos dependiendo de hora, fecha y/o Tema

Alcance de su proyecto:

Esto en el futuro se puede utilizar para las compañías, departamentos de gobierno (Secretarias) o uso para cualquier persona para que estén programados en días y horas que avisaran al usuario cuando van a alcanzar su límite de entrega si es en caso de las compañías(cronometro o alarma), además con una implementación antihackeo. Pero esto se agregara a futuro y dependiendo de lo aprendido durante mi carrera.

Introducción:

Durante lo visto en la materia de Estructuras y Datos 1 hemos desarrollado u observado la forma de almacenamiento de datos que han utilizado compañías como Facebook, Windows y en lo que me interesa los videojuegos. En este caso me intereso mucho el hacer este proyecto.

Al principio al platicar con amigos sobre lo que sería nuestros proyectos este empezó como una pokedex (enciclopedia de Pokemones) ya que en la aplicación llama Pokemon Go no se tiene una descripción y/o evoluciones de estas criaturas a como los tiene los juegos así nombrados, en este caso seria los primeros 150.

Pero este cambio al platicar con mi madre que trabaja en la Policía Federal actualmente Guardia Nacional en donde me estaba diciendo que la manera de ordenar documentos o su base de datos es por medio de la aplicación de Excel en donde está clasificado por fecha, asunto y documentos. En esta forma de uso para un departamento de gobierno o compañía es muy obsoleta para esta ya en un caso de emergencia seria de una forma muy lenta y obsoleta a lo que nuestra tecnología nos puede permitir.

Al final de lo platicado y reflexionado he decido hacer un programa en donde se podrá manejar por medio de apartados en donde se clasificara de lo visto o lo que desee el usuario en donde dependiendo de sus usos.

Desarrollo:

Lenguaje a utilizar: C

En este proyecto se utilizara varios temas vistos en nuestras clases como los arreglos y los apuntadores pero en lo visto en lo que se va a realizar se ocupara precisamente los datos abstractos, esto nos servirá para la clasificación de datos o documentos y más cuando será cuando se agregue o se tenga que ver próximamente.

- ***Definición:***

❖ *Arreglo:*

Es un conjunto de datos finito y del mismo tipo. En realidad funciona como cualquier variable cualquiera, excepto que en lugar de almacenar un solo valor, guarda algunos valores. Pueden ser unidimensionales o multidimensionales. Los arreglos nos permiten hacer un conjunto de operaciones para manipular los datos guardados en ellos, estas operaciones son: ordenar, buscar, insertar, eliminar, modificar entre otras.

❖ *Apuntador:*

Es una variable que contiene una dirección de memoria, la cual corresponderá a un dato o a una variable que contiene el dato. Cada variable que se utiliza en una aplicación ocupa una o varias posiciones de memoria. Estas posiciones de memoria se accedan por medio de una dirección

❖ *Datos abstractos:*

Un tipo de dato abstracto (TDA) es un conjunto de datos u objetos creado de manera personalizada por un programador para un fin específico. Un TDA es una abstracción que permite modelar las características de un elemento en particular.

Un tipo de dato abstracto se puede manipular de forma similar a los tipos de datos que están predefinidos dentro del lenguaje de programación, encapsulando más información, según se requiera.

La implementación de un tipo de dato abstracto depende directamente del lenguaje de programación que se utilice. En lenguaje C los tipos de dato abstracto se crean mediante las estructuras (struct).

- **Algoritmo:**
- **Diagrama de flujo:**
- **Pseudocódigo:**
- **Código**

```c
#include <stdio.h>    /* Funciones principales */
#include <string.h>   /* operaciones de manipulación de memoria */
#include <stdlib.h>   /* Gestión de memoria dinámica, control de procesos, etc */
#include <locale.h>   /* configuración regional actual */

#define MAX  1
#define VALOR_CENTINELA -1

struct informacion {
    int codigo;
    char nombre[MAX];
    char descripcion[MAX];
    char archivo[MAX];

};

typedef struct informacion Informacion;

void menuPrincipal();
void menuInsertar();
void menuBuscar();
void menuEliminar();
void menuMostrar();
void menuModificar();
void menuEliminarFisica();

/* Funciones para manejar el archivo directamente */
Informacion *obtenerInformaciones(int *n); /* Obtiene un vector dinámico de archivos */
char existeInformacion(int codigoInformacion, Informacion *informacion);
char insertarInformacion(Informacion informacion);
char eliminarInformacion(int codigoInformacion);
char eliminacionFisica();
char modificarInformacion(Informacion informacion);
char guardarReporte(); /* Genera un archivo TXT */

/* Función de lectura de cadenas */
int leecad(char *cad, int n);

/* Titular del programa */
void tituloPrincipal();

char linea[MAX];

int main()
{
    setlocale(LC_ALL, "spanish"); /* Permite imprimir caracteres con tilde */
    menuPrincipal();

    return 0;
}

void menuPrincipal()
{
    char repite = 1;
    int opcion = -1;
    /* Cuando el usuario ingresa texto en lugar de ingresar una opción. El programa no modifica
    el valor de opcion. En ese caso, no se debe de ingresar a ninguno de los case, por eso se está
    inicializando la variable opcion con un valor que no permita ejecutar ningún case. Simplemente,
    volver a interar y pedir nuevamente la opción. */

    do {
        system("cls");

        tituloPrincipal();

        printf("\n\t\t\t\tMENU PRINCIPAL\n");
        printf("\n\t\t[1]. Insertar nuevo registro\n");
        printf("\t\t[2]. Mostrar listado de registros\n");
        printf("\t\t[3]. Eliminar un registro\n");
        printf("\t\t[4]. Buscar registro por numero o fecha\n");
        printf("\t\t[5]. Modificar un registro\n");
        printf("\t\t[6]. Eliminación física de registros\n");
        printf("\t\t[7]. Salir\n");
        printf("\n\t\tIngrese su opción: [ ]\b\b");

        /* Lectura segura de un entero */
        leecad(linea, MAX);
        sscanf(linea, "%d", &opcion);
```

```
        switch (opcion) {

            case 1:
                menuInsertar();
                break;

            case 2:
                menuMostrar();
                break;

            case 3:
                menuEliminar();
                break;

            case 4:
                menuBuscar();
                break;

            case 5:
                menuModificar();
                break;

            case 6:
                menuEliminarFisica();
                break;

            case 7:
                repite = 0;
                break;
        }

    } while (repite);
}

void menuInsertar(){

    Informacion informacion;
    int codigoInformacion = 0;
    char repite = 1;
    char respuesta[MAX];

    do {
        system("cls");
        tituloPrincipal();
        printf("\n\t\t\t==> INSERTAR INFORMACION <==\n");

        /* Se pide el código del producto a insertar */
        printf("\n\tFecha o numero: ");
        leecad(linea, MAX);
        sscanf(linea, "%s", &codigoInformacion);

        /* Se verifica que el producto no haya sido almacenado anteriormente */
        if (!existeInformacion(codigoInformacion, &informacion)){

            informacion.codigo = codigoInformacion;

            /* Se piden los demás datos del producto a insertar */
            printf("\tNombre del registro: ");
            leecad(informacion.nombre, MAX);

            printf("\tDescripcion: ");
            leecad(informacion.descripcion, MAX);

            printf("\tArchivo: ");
            leecad(informacion.archivo, MAX);

            if (insertarInformacion(informacion)) {
                printf("\n\tEl registro fue insertado correctamente\n");

            } else {
                printf("\n\tOcurrió un error al intentar insertar el registro\n");
                printf("\tInténtelo mas tarde\n");
            }
        } else {
            /* El registro ya existe, no puede ser insertado. */
            printf("\n\tLa información de fecha o numero %f ya existe.\n", codigoInformacion);
            printf("\tNo puede ingresar dos registros distintos con el mismo código.\n");
        }

        printf("\n\tDesea seguir ingresando registros? [S/N]: ");
        leecad(respuesta, MAX);

        if (!(strcmp(respuesta, "S") == 0 || strcmp(respuesta, "s") == 0)) {
            repite = 0;
        }

    } while (repite);
}

void menuBuscar(){
    Informacion informacion;
    int codigoInformacion;
    char repite = 1;
    char respuesta[MAX];

    do {
        system("cls");
        tituloPrincipal();
        printf("\n\t\t\t==> BUSCAR REGISTRO POR FECHA O NUMERO <==\n");

        /* Se pide el código del producto a buscar */
        printf("\n\tFecha o numero del registro: ");
        leecad(linea, MAX);
        sscanf(linea, "%s", &codigoInformacion);

        /* Se verifica que el registro a buscar, exista */
        if (existeInformacion(codigoInformacion, &informacion)) {

            /* Se muestran los datos del producto */
            printf("\n\tFecha o numero del registro: %s\n", informacion.codigo);
            printf("\tNombre del registro: %s\n", informacion.nombre);
            printf("\tDescripcion del registro: %s\n", informacion.descripcion);
            printf("\tArchivo del registro: %s\n", informacion.archivo);

        } else {
            /* El producto no existe */
            printf("\n\tEl registro de la fecha o numero %s no existe.\n", codigoInformacion);
        }

        printf("\n\tDesea seguir buscando algún producto? [S/N]: ");
        leecad(respuesta, MAX);

        if (!(strcmp(respuesta, "S") == 0 || strcmp(respuesta, "s") == 0)) {
            repite = 0;
        }

    } while (repite);
}

void menuEliminar()
{
    Informacion informacion;
    int codigoInformacion;
    char repite = 1;
    char respuesta[MAX];

    do {
        system("cls");
        tituloPrincipal();
        printf("\n\t\t\t==> ELIMINAR REGISTRO POR FECHA O NUMERO <==\n");

        /* Se pide la fecha o el numero del registro a eliminar */
        printf("\n\tFecha o numero del registro: ");
        leecad(linea, MAX);
        sscanf(linea, "%s", &codigoInformacion);

        /* Se verifica que el registro a buscar, exista */
        if (existeInformacion(codigoInformacion, &informacion)) {

            /* Se muestran los datos del registro */
            printf("\n\tFecha o numero del registro: %d\n", informacion.codigo);
            printf("\tNombre del registro: %s\n", informacion.nombre);
            printf("\tDescripcion del registro: %s\n", informacion.descripcion);
            printf("\tArchivo del registro: %s\n", informacion.archivo);

            printf("\n\tSeguro que desea eliminar el registro? [S/N]: ");
            leecad(respuesta, MAX);
            if (strcmp(respuesta, "S") == 0 || strcmp(respuesta, "s") == 0) {
```

```
                if (eliminarInformacion(codigoInformacion)) {
                    printf("\n\tRegistro eliminado satisfactoriamente.\n");
                } else {
                    printf("\n\tEl registro no pudo ser eliminado\n");
                }
            }

        } else {
            /* El registro no existe */
            printf("\n\tEl registro de la fecha o numero %d no existe.\n", codigoInformacion);
        }

        printf("\n\tDesea eliminar otro registro? [S/N]: ");
        leecad(respuesta, MAX);

        if (!(strcmp(respuesta, "S") == 0 || strcmp(respuesta, "s") == 0)) {
            repite = 0;
        }

    } while (repite);
}

void menuMostrar()
{
    Informacion *informaciones;
    int numeroInformaciones;
    int i;
    char respuesta[MAX];

    system("cls");
    tituloPrincipal();
    informaciones = obtenerInformaciones(&numeroInformaciones); /* Retorna un vector dinámico de la informacion */

    if (numeroInformaciones == 0) {
        printf("\n\tEl archivo está vacio!!\n");
        system("pause>nul");

    } else {
        printf("\n\t\t    ==> LISTADO DE REGISTROS INGRESADOS <==\n");
        printf(" ------------------------------------------------------------------------------\n");
        printf("%8s\t%-20s%15s%15s\n", "FECHA O NUMERO", "NOMBRE DEL REGISTRO", "DESCRIPCION", "ARCHIVO");
        printf(" ------------------------------------------------------------------------------\n");

        /* Se recorre el vector dinámico de productos */
        for (i = 0; i < numeroInformaciones; i++) {
            if (informaciones[i].codigo != VALOR_CENTINELA) {
                printf("%7s \t%-20.20s%15s%15s\n", informaciones[i].codigo, informaciones[i].nombre, informaciones[i].descripcion, informac
            }
        }

        printf("\n\tDesea guardar el reporte en un archivo de texto? [S/N]: ");
        leecad(respuesta, MAX);

        if (strcmp(respuesta, "S") == 0 || strcmp(respuesta, "s") == 0) {
            if (guardarReporte()) {
                printf("\n\tEl reporte fue guardado con éxito\n");
            } else {
                printf("\n\tOcurrió un error al guardar el reporte\n");
            }

            system("pause>nul");
        }
    }
}

void menuModificar()
{
    Informacion informacion;
    int codigoInformacion;
    char repite = 1;
    char respuesta[MAX];

    do {
        system("cls");
        tituloPrincipal();
        printf("\n\t\t\t==> MODIFICAR REGISTRO POR FECHA O NUMERO <==\n");

        /* Se pide la fecha o el numero del registro a modificar */
        printf("\n\tFecha o numero del registro: ");
        leecad(linea, MAX);
        sscanf(linea, "%s", &codigoInformacion);

        /* Se verifica que el producto a buscar exista */
        if (existeInformacion(codigoInformacion, &informacion)) {

            /* Se muestran los datos del registro */
            printf("\n\tNombre del registro: %s\n", informacion.nombre);
            printf("\tDescripcion: %s\n", informacion.descripcion);
            printf("\tArchivo: %s\n", informacion.archivo);

            printf("\n\tElija los datos a modificar\n");

            /* Modificación del nombre del registro */
            printf("\n\tNombre del registro actual: %s\n", informacion.nombre);
            printf("\tDesea modificar el nombre del registro? [S/N]: ");
            leecad(respuesta, MAX);
            if (strcmp(respuesta, "S") == 0 || strcmp(respuesta, "s") == 0) {
                printf("\tNuevo nombre del registro: ");
                leecad(informacion.nombre, MAX);
            }

            /* Modificación de la descripcion del registro */
            printf("\n\tDescripcion: %s\n", informacion.descripcion);
            printf("\tDesea modificar la descripcion? [S/N]: ");
            leecad(respuesta, MAX);
            if (strcmp(respuesta, "S") == 0 || strcmp(respuesta, "s") == 0) {
                printf("\tNueva descripcion del registro: ");
                leecad(linea, MAX);
                sscanf(linea, "%s", &informacion.descripcion);
            }

            /* Modificación del archivo del registro */
            printf("\n\tArchivo del registro actual: %s\n", informacion.archivo);
            printf("\tDesea modificar el archivo del registro? [S/N]: ");
            leecad(respuesta, MAX);
            if (strcmp(respuesta, "S") == 0 || strcmp(respuesta, "s") == 0) {
                printf("\tNuevo archivo del registro: ");
                leecad(linea, MAX);
                sscanf(linea, "%s", &informacion.archivo);
            }

            printf("\n\tEstá seguro que desea modificar los datos del registro? [S/N]: ");
            leecad(respuesta, MAX);

            if (strcmp(respuesta, "S") == 0 || strcmp(respuesta, "s") == 0) {
                /* Se modifica el registro en el archivo */
                if (modificarInformacion(informacion)) {
                    printf("\n\tEl registro fue modificado correctamente\n");

                } else {
                    printf("\n\tOcurrió un error al intentar modificar el registro\n");
                    printf("\tInténtelo mas tarde\n");
                }
            }
        } else {
            /* El registro no existe */
            printf("\n\tEl registro de la fecha o del numero %s no existe.\n", codigoInformacion);
        }

        printf("\n\tDesea modificar algún otro registro? [S/N]: ");
        leecad(respuesta, MAX);

        if (!(strcmp(respuesta, "S") == 0 || strcmp(respuesta, "s") == 0)) {
            repite = 0;
        }

    } while (repite);
}

void menuEliminarFisica(){

    char respuesta[MAX];

    system("cls");
    tituloPrincipal();
    printf("\n\t\t==> ELIMINAR FÍSICAMENTE REGISTROS DEL ARCHIVO <==\n");
```

```
    /* Se pide el código del registro a eliminar */
    printf("\n\tSeguro que desea proceder con la eliminación física? [S/N]: ");
    leecad(respuesta, MAX);

    if (strcmp(respuesta, "S") == 0 || strcmp(respuesta, "s") == 0) {
        if (eliminacionFisica()) {
            printf("\n\tLa eliminación física se realizó con éxito.\n");
        } else {
            printf("\n\tOcurrió algún error en la eliminación física.\n");
        }

        system("pause>nul");
    }
}

Informacion *obtenerInformacion(int *n){

    FILE *archivo;
    Informacion informacion;
    Informacion *informaciones; /* Vector dinámico de informacion */
    int i;

    /* Abre el archivo en modo lectura */
    archivo = fopen("informacion.dat", "rb");

    if (archivo == NULL) { /* Si no se pudo abrir el archivo, el valor de archivo es NULL */
        *n = 0; /* No se pudo abrir. Se considera n  */
        informaciones = NULL;

    } else {

        fseek(archivo, 0, SEEK_END); /* Posiciona el cursor al final del archivo */
        *n = ftell(archivo) / sizeof(Informacion); /* # de registros almacenados en el archivo. (# de registros) */
        informaciones = (Informacion *)malloc((*n) * sizeof(Informacion)); /* Se reserva memoria para todos los registros almacenados en el

        /* Se recorre el archivo secuencialmente */
        fseek(archivo, 0, SEEK_SET); /* Posiciona el cursor al principio del archivo */
        fread(&informacion, sizeof(informacion), 1, archivo);
        i = 0;
        while (!feof(archivo)) {
            informaciones[i++] = informacion;
            fread(&informacion, sizeof(informacion), 1, archivo);
        }

        /* Cierra el archivo */
        fclose(archivo);
    }

    return informaciones;
}

char existeInformacion(int codigoInformacion, Informacion *informacion)
{
    FILE *archivo;
    char existe;

    /* Abre el archivo en modo lectura */
    archivo = fopen("informacion.dat", "rb");

    if (archivo == NULL) { /* Si no se pudo abrir el archivo, el valor de archivo es NULL */
        existe = 0;

    } else {
        existe = 0;

        /* Se busca el registro cuyo código coincida con codigoInformacion */
        fread(&(*informacion), sizeof(*informacion), 1, archivo);
        while (!feof(archivo)) {
            if ((*informacion).codigo == codigoInformacion) {
                existe = 1;
                break;
            }
            fread(&(*informacion), sizeof(*informacion), 1, archivo);
        }

        /* Cierra el archivo */
        fclose(archivo);
    }

    return existe;
}

char insertarInformacion(Informacion informacion)
{
    FILE *archivo;
    char insercion;

    /* Abre el archivo para agregar datos al final */
    archivo = fopen("Informacion.dat", "ab");    /* Añade datos al final. Si el archivo no existe, es creado */

    if (archivo == NULL) { /* Si no se pudo abrir el archivo, el valor de archivo es NULL */
        insercion = 0;

    } else {
        fwrite(&informacion, sizeof(informacion), 1, archivo);
        insercion = 1;

        /* Cierra el archivo */
        fclose(archivo);
    }

    return insercion;
}

/* ELiminación lógica de un registro */
char eliminarInformacion(int codigoInformacion)
{
    FILE *archivo;
    FILE *auxiliar;
    Informacion informacion;
    char elimina;

    /* Abre el archivo para leer */
    archivo = fopen("Informacion.dat", "r+b");    /* Modo lectura/escritura. Si el archivo no existe, es creado */

    if (archivo == NULL) { /* Si no se pudo abrir el archivo, el valor de archivo es NULL */
        elimina = 0;
    } else {
        /* Se busca el registro que se quiere borrar. Cuando se encuentra, se sitúa en esa posición mediante la
        función fseek y luego se modifica el campo clave de ese registro mediante algún valor centinela, eso se logra
        con fwrite. Hasta allí se ha logrado una eliminación LÓGICA. Porque el registro sigue ocupando espacio en el archivo físico */

        elimina = 0;
        fread(&informacion, sizeof(informacion), 1, archivo);
        while (!feof(archivo)) {
            if (informacion.codigo == codigoInformacion) {
                fseek(archivo, ftell(archivo) - sizeof(informacion), SEEK_SET);
                informacion.codigo = VALOR_CENTINELA;
                fwrite(&informacion, sizeof(informacion), 1, archivo);
                elimina = 1;
                break;
            }
            fread(&informacion, sizeof(informacion), 1, archivo);
        }

        /* Cierra el archivo */
        fclose(archivo);
    }

    return elimina;
}

char eliminacionFisica()
{
    FILE *archivo;
    FILE *temporal;
    Informacion informacion;
    char elimina = 0;

    archivo = fopen("Informacion.dat", "rb");
    temporal = fopen("temporal.dat", "wb");

    if (archivo == NULL || temporal == NULL) {
        elimina = 0;
    } else {
        /* Se copia en el archivo temporal los registros válidos */
```

```
        fread(&informacion, sizeof(informacion), 1, archivo);
        while (!feof(archivo)) {
            if (informacion.codigo != VALOR_CENTINELA) {
                fwrite(&informacion, sizeof(informacion), 1, temporal);
            }
            fread(&informacion, sizeof(informacion), 1, archivo);
        }
        /* Se cierran los archivos antes de borrar y renombrar */
        fclose(archivo);
        fclose(temporal);

        remove("Informacion.dat");
        rename("temporal.dat", "Informacion.dat");

        elimina = 1;
    }

    return elimina;
}

char modificarInformacion(Informacion informacion)
{
    FILE *archivo;
    char modifica;
    Informacion informacion2;

    /* Abre el archivo para lectura/escritura */
    archivo = fopen("Informacion.dat", "rb+");

    if (archivo == NULL) { /* Si no se pudo abrir el archivo, el valor de archivo es NULL */
        modifica = 0;

    } else {
        modifica = 0;
        fread(&informacion2, sizeof(informacion2), 1, archivo);
        while (!feof(archivo)) {
            if (informacion2.codigo == informacion.codigo) {
                fseek(archivo, ftell(archivo) - sizeof(informacion), SEEK_SET);
                fwrite(&informacion, sizeof(informacion), 1, archivo);
                modifica = 1;
                break;
            }
            fread(&informacion2, sizeof(informacion2), 1, archivo);
        }

        fclose(archivo);
    }

    /* Cierra el archivo */
    return modifica;
}

char guardarReporte()
{
    FILE *archivo;
    char guardado;
    Informacion *informaciones;
    int numeroInformaciones;
    int i;

    informaciones = obtenerInformaciones(&numeroInformaciones); /* Retorna un vector dinámico de registros */

    if (numeroInformaciones == 0) {
        guardado = 0;

    } else {
        /* Abre el archivo en modo texto para escritura */
        archivo = fopen("reporte.txt", "w");

        if (archivo == NULL) { /* Si no se pudo abrir el archivo, el valor de archivo es NULL */
            guardado = 0;

        } else {
            fprintf(archivo, "\n\t\t    ==> LISTADO DE PRODUCTOS REGISTRADOS <==\n");
            fprintf(archivo, " ------------------------------------------------------------------------------\n");
            fprintf(archivo, "%8s\t%-20s%15s%15s\n", "FECHA O NUMERO", "NOMBRE DEL REGISTRO", "DESCRIPCION", "ARCHIVO");
            fprintf(archivo, " ------------------------------------------------------------------------------\n");
```

```
            /* Se recorre el vector dinámico de registros */
            for (i = 0; i < numeroInformaciones; i++) {
                if (informaciones[i].codigo != VALOR_CENTINELA) {
                    fprintf(archivo, "%7s \t%-20.20s%15s%15s\n", informaciones[i].codigo, informaciones[i].nombre, informaciones[i].descrip
                }
            }

            guardado = 1;

            /* Cierra el archivo */
            fclose(archivo);
        }
    }

    return guardado;
}

int leecad(char *cad, int n)
{
    int i, c;

    /* Hay que verificar si el buffer está limpio o si hay un '\n'
      dejado por scanf y, en ese caso, limpiarlo:
    */

    /* 1 COMPROBACIÓN DE DATOS INICIALES EN EL BUFFER */

    /* Empezamos leyendo el primer caracter que haya en la entrada. Si es
      EOF, significa que no hay nada por leer, así que cerramos la cadena,
      dejándola "vacía" y salimos de la función retornando un valor de 0
      o falso, para indicar que hubo un error */
    c = getchar();
    if (c == EOF) {
        cad[0] = '\0';
        return 0;
    }

    /* Si el valor leído es '\n', significa que había un caractar de nueva línea
    dejado por un scanf o función similar. Simplemente inicializamos i a 0,
```

```
      para indicar que los siguientes caracteres que leamos iremos asignando a
      partir del primer caracter de la cadena. */
      if (c == '\n') {
          i = 0;
      } else {
      /* Si no había un '\n', significa que el caracter que leímos es el primer
        caracter de la cadena introducida. En este caso, lo guardamos en la
        posición 0 de cad, e inicializamos i a 1, porque en este caso, como ya
        tenemos el primer caracter de la cadena, continuaremos agregando
        caracteres a partir del segundo.

      */
          cad[0] = c;
          i = 1;
      }

      /* 2. LECTURA DE LA CADENA */

      /* El for empieza con un ; porque estamos omitiendo la inicialización del contador,
      ya que fue inicializado en el punto anterior.
      Este código lee un caracter a la vez,lo agrega a cad, y se repite hasta que
      se encuentre un fin de línea, fin de archivo, o haya leído la cantidad máxima
      de caracteres que se le indicó. Luego, cierra la cadena agregando un '\0'
      al final. Todo esto es muy similar a la forma en que los compiladores suelen
      implementar la función fgets, sólo que en lugar de getchar usan getc o fgetc
      */
      for (; i < n - 1 && (c = getchar()) != EOF && c != '\n'; i++) {
          cad[i] = c;
      }
      cad[i] = '\0';

      /*3. LIMPIEZA DEL BUFFER */

      /* Finalmente limpiamos el buffer si es necesario */
      if (c != '\n' && c != EOF) /* es un caracter */
          while ((c = getchar()) != '\n' && c != EOF);

      /* La variable c contiene el último caracter leído. Recordemos que había 3 formas
      de salir del for: que hayamos encontrando un '\n', un EOF, o que hayamos llegado
```

```
      de salir del for: que hayamos encontrando un '\n', un EOF, o que hayamos llegado
      al máximo de caracteres que debemos leer. Si se da cualquiera de los dos
      primeros casos, significa que leímos todo lo que había en el buffer, por lo que
      no hay nada que limpiar. En el tercer caso, el usuario escribió más caracteres
      de los debidos, que aún están en el buffer, por lo que hay que quitarlos, para
      lo cual usamos el método que vimos poco más arriba
      */

      return i;
  }

  void tituloPrincipal()
  {
      int i;
      printf("\n      ==========================================================================\n");
      printf("\t\t\t     PROYECTO FINAL\n");
      printf("\t   Base de datos: Creación, reportes, eliminación, búsqueda y actualización\n");
      printf("\t\t     Arroyo Chavarría Jose Luis\n");
      printf("      ==========================================================================\n");

      i = 0;
      putchar('\n');
      for (; i < 80; i++) {
          putchar('_');
      }
  }
```

Resultados:

Conclusiones:

Referencias:

- https://es.wikipedia.org/wiki/Memoria_dinámica_(programación)
- https://www.ecured.cu/Pila_(Estructura_de_datos)
- http://www.utn.edu.ec/reduca/programacion/arreglos/definiciones1.html
- http://www.utm.mx/~mgarcia/PE7(Apuntadores).pdf
- Guía práctica de estudio 03. Manual de prácticas del Laboratorio de Estructuras de datos y algoritmos I. Tipo de dato abstracto. M.C. Edgar E. García Cano e Ing. Jorge A. Solano Gálvez. UNAM. 2017. Pags. 25 - 32