	<h2>Proyecto Final</h2>	
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 (Secretarías) 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:**

Entradas:

- Registro de archivos para que el usuario desee poner
- Modificar, mostrar, buscar y eliminar los registros guardados

Salida:

- Base de datos donde están los registros guardados que el usuario podrá utilizar dependiendo lo que desee.

Proceso:

- Ubicar por medio de un código los archivos a guardar y hacer procesos para ejecutar las acciones que podrá el usuario utilizar.

- **Diagrama de flujo:**

- **Pseudocódigo:**

Variable:	Descripción:	Tipo:
O	Opción a elegir	Real
C	Código	Caracteres
F	Fecha o numero	Entero
R	Nombre del Registro	Información
D	Descripción del registro	Información
A	Archivo	Información

1. Inicio

2. Leer O
3. Según O hacer
4. Caso 1
5. Leer C, F, R, D y A
6. Escribir C, F, R, D y A
7. Caso 2
8. Escribir C, F, R, D y A
9. Caso 3
10. Leer C
11. Si C existe entonces
- 12.
13. Borrar C, F, R, D y A
14. Escribir Registro borrado
15. Sino
16. Escribir Registro no encontrando
17. FinSi
18. Caso 4
19. Leer C
20. Escribir C, F, R, D y A
21. Caso 5
22. Leer C
23. Si C existe entonces
24. Leer F
25. Si F entonces
26. Escribir F
27. Sino
28. Escribir Fecha o número no modificado
29. FinSi
30. Leer R
31. Si R entonces
32. Escribir R

- 33. Sino
- 34. Escribir Nombre del registro no modificado
- 35. FinSi
- 36. Leer D
- 37. Si D entonces
- 38. Escribir D
- 39. Sino
- 40. Escribir Descripción no modificado
- 41. FinSi
- 42. Leer A
- 43. Si A entonces
- 44. Escribir A
- 45. Sino
- 46. Escribir Archivo no modificado
- 47. FinSi
- 48. Sino
- 49. Escribir Cambios no realizados o Código no encontrado
- 50. FinSi
- 51. FinSegun
- 52. Fin

- **Código**

```

1 #include <stdio.h> /* Funciones principales */
2 #include <string.h> /* operaciones de manipulación de memoria */
3 #include <stdlib.h> /* Gestión de memoria dinámica, control de procesos, etc */
4 #include <locale.h> /* configuración regional actual */
5
6 #define MAX 1
7 #define VALOR_CENTINELA -1
8
9 struct informacion {
10     int codigo;
11     char nombre[MAX];
12     char descripcion[MAX];
13     char archivo[MAX];
14 };
15
16 typedef struct informacion Informacion;
17
18 void menuPrincipal();
19 void menuInsertar();
20 void menuBuscar();
21 void menuEliminar();
22 void menuMostrar();
23 void menuModificar();
24 void menuEliminarFisica();
25
26 /* Funciones para manejar el archivo directamente */
27 Informacion *obtenerInformaciones(int *n); /* Obtiene un vector dinámico de archivos */
28 char existeInformacion(int codigoInformacion, Informacion *informacion);
29 char insertarInformacion(Informacion informacion);
30 char eliminarInformacion(int codigoInformacion);
31 char eliminacionFisica();
32 char modificarInformacion(Informacion informacion);
33 char guardarReporte(); /* Genera un archivo TXT */
34
35 /* Función de lectura de cadenas */
36 int leecad(char *cad, int n);
37
38 /* Titular del programa */

```

```

40 void tituloPrincipal();
41
42 char linea[MAX];
43
44 int main()
45 {
46     setlocale(LC_ALL, "spanish"); /* Permite imprimir caracteres con tilde */
47     menuPrincipal();
48
49     return 0;
50 }
51
52 void menuPrincipal()
53 {
54     char repite = 1;
55     int opcion = -1;
56     /* Cuando el usuario ingresa texto en lugar de ingresar una opción. El programa no modifica
57     el valor de opción. En ese caso, no se debe de ingresar a ninguno de los case, por eso se está
58     inicializando la variable opción con un valor que no permita ejecutar ningún case. Simplemente,
59     volver a interactuar y pedir nuevamente la opción. */
60
61     do {
62         system("cls");
63
64         tituloPrincipal();
65
66         printf("\n\t\t\t\t\tMENU PRINCIPAL\n");
67         printf("\n\t\t\t\t\t1. Insertar nuevo registro\n");
68         printf("\n\t\t\t\t\t2. Mostrar listado de registros\n");
69         printf("\n\t\t\t\t\t3. Eliminar un registro\n");
70         printf("\n\t\t\t\t\t4. Buscar registro por numero o fecha\n");
71         printf("\n\t\t\t\t\t5. Modificar un registro\n");
72         printf("\n\t\t\t\t\t6. Eliminación física de registros\n");
73         printf("\n\t\t\t\t\t7. Salir\n");
74         printf("\n\t\t\t\t\tIngrese su opción: [ ]\b\b");
75
76         /* Lectura segura de un entero */
77         leecad(linea, MAX);
78         sscanf(linea, "%d", &opcion);

```

```

80         switch (opcion) {
81
82             case 1:
83                 menuInsertar();
84                 break;
85
86             case 2:
87                 menuMostrar();
88                 break;
89
90             case 3:
91                 menuEliminar();
92                 break;
93
94             case 4:
95                 menuBuscar();
96                 break;
97
98             case 5:
99                 menuModificar();
100                 break;
101
102             case 6:
103                 menuEliminarFisica();
104                 break;
105
106             case 7:
107                 repite = 0;
108                 break;
109         }
110     } while (repite);
111 }
112
113 void menuInsertar(){
114     Informacion informacion;
115     int codigoInformacion = 0;
116     char repite = 1;

```

```

119     char respuesta[MAX];
120
121     do {
122         system("cls");
123         tituloPrincipal();
124         printf("\n\t\t\t\t\tINSERTAR INFORMACION <==\n");
125
126         /* Se pide el código del producto a insertar */
127         printf("\n\t\t\t\t\tFecha o numero: ");
128         leecad(linea, MAX);
129         sscanf(linea, "%s", &codigoInformacion);
130
131         /* Se verifica que el producto no haya sido almacenado anteriormente */
132         if (!existeInformacion(codigoInformacion, &informacion)){
133
134             informacion.codigo = codigoInformacion;
135
136             /* Se piden los demás datos del producto a insertar */
137             printf("\n\t\t\t\t\tNombre del registro: ");
138             leecad(informacion.nombre, MAX);
139
140             printf("\n\t\t\t\t\tDescripción: ");
141             leecad(informacion.descripcion, MAX);
142
143             printf("\n\t\t\t\t\tArchivo: ");
144             leecad(informacion.archivo, MAX);
145
146             if (insertarInformacion(informacion)) {
147                 printf("\n\t\t\t\t\tEl registro fue insertado correctamente\n");
148             }
149             else {
150                 printf("\n\t\t\t\t\tOcurrió un error al intentar insertar el registro\n");
151                 printf("\n\t\t\t\t\tInténtelo mas tarde\n");
152             }
153         } else {
154             /* El registro ya existe, no puede ser insertado. */
155             printf("\n\t\t\t\t\tLa información de fecha o numero %f ya existe.\n", codigoInformacion);
156             printf("\n\t\t\t\t\tNo puede ingresar dos registros distintos con el mismo código.\n");
157         }

```



```

316     printf("\n\tFecha o numero del registro: ");
317     leecad(linea, MAX);
318     sscanf(linea, "%s", &codigoInformacion);
319
320     /* Se verifica que el producto a buscar exista */
321     if (existeInformacion(codigoInformacion, &informacion)) {
322
323         /* Se muestran los datos del registro */
324         printf("\n\tNombre del registro: %s\n", informacion.nombre);
325         printf("\tDescripcion: %s\n", informacion.descripcion);
326         printf("\tArchivo: %s\n", informacion.archivo);
327
328         printf("\n\tElija los datos a modificar\n");
329
330         /* Modificación del nombre del registro */
331         printf("\n\tNombre del registro actual: %s\n", informacion.nombre);
332         printf("\tDesea modificar el nombre del registro? [S/N]: ");
333         leecad(respuesta, MAX);
334         if (strcmp(respuesta, "S") == 0 || strcmp(respuesta, "s") == 0) {
335             printf("\tNuevo nombre del registro: ");
336             leecad(informacion.nombre, MAX);
337         }
338
339         /* Modificación de la descripción del registro */
340         printf("\n\tDescripción: %s\n", informacion.descripcion);
341         printf("\tDesea modificar la descripción? [S/N]: ");
342         leecad(respuesta, MAX);
343         if (strcmp(respuesta, "S") == 0 || strcmp(respuesta, "s") == 0) {
344             printf("\tNueva descripción del registro: ");
345             leecad(linea, MAX);
346             sscanf(linea, "%s", &informacion.descripcion);
347         }
348
349         /* Modificación del archivo del registro */
350         printf("\n\tArchivo del registro actual: %s\n", informacion.archivo);
351         printf("\tDesea modificar el archivo del registro? [S/N]: ");
352         leecad(respuesta, MAX);
353         if (strcmp(respuesta, "S") == 0 || strcmp(respuesta, "s") == 0) {
354             printf("\tNuevo archivo del registro: ");

```

```

355             leecad(linea, MAX);
356             sscanf(linea, "%s", &informacion.archivo);
357         }
358
359         printf("\n\t¿Está seguro que desea modificar los datos del registro? [S/N]: ");
360         leecad(respuesta, MAX);
361
362         if (strcmp(respuesta, "S") == 0 || strcmp(respuesta, "s") == 0) {
363             /* Se modifica el registro en el archivo */
364             if (modificarInformacion(informacion)) {
365                 printf("\n\tEl registro fue modificado correctamente\n");
366             } else {
367                 printf("\n\tOcurrió un error al intentar modificar el registro\n");
368                 printf("\tInténtelo mas tarde\n");
369             }
370         } else {
371             /* El registro no existe */
372             printf("\n\tEl registro de la fecha o del numero %s no existe.\n", codigoInformacion);
373         }
374
375         printf("\n\tDesea modificar algún otro registro? [S/N]: ");
376         leecad(respuesta, MAX);
377
378         if (!strcmp(respuesta, "S") == 0 || strcmp(respuesta, "s") == 0) {
379             repite = 0;
380         } while (repite);
381     }
382
383     void menuEliminarFisica(){
384
385         char respuesta[MAX];
386
387         system("cls");
388         tituloPrincipal();
389         printf("\n\t\t== ELIMINAR FÍSICAMENTE REGISTROS DEL ARCHIVO <==\n");

```

```

395     /* Se pide el código del registro a eliminar */
396     printf("\n\t¿Seguro que desea proceder con la eliminación física? [S/N]: ");
397     leecad(respuesta, MAX);
398
399     if (strcmp(respuesta, "S") == 0 || strcmp(respuesta, "s") == 0) {
400         if (eliminarFisica()) {
401             printf("\n\tLa eliminación física se realizó con éxito.\n");
402         } else {
403             printf("\n\tOcurrió algún error en la eliminación física.\n");
404         }
405
406         system("pause\n");
407     }
408 }
409
410 Informacion *obtenerInformacion(int n){
411     FILE *archivo;
412     Informacion informacion;
413     Informacion *informaciones; /* Vector dinámico de informacion */
414     int i;
415
416     /* Abre el archivo en modo lectura */
417     archivo = fopen("informacion.dat", "rb");
418
419     if (archivo == NULL) { /* Si no se pudo abrir el archivo, el valor de archivo es NULL */
420         "n = 0; /* No se pudo abrir. Se considera n = 0 */
421         informaciones = NULL;
422     } else {
423
424         fseek(archivo, 0, SEEK_END); /* Posiciona el cursor al final del archivo */
425         "n = ftell(archivo) / sizeof(Informacion); /* # de registros almacenados en el archivo. (# de registros) */
426         informaciones = (Informacion *)malloc("n * sizeof(Informacion)); /* Se reserva memoria para todos los registros almacenados en el
427
428         /* Se recorre el archivo secuencialmente */
429         fseek(archivo, 0, SEEK_SET); /* Posiciona el cursor al principio del archivo */
430         fread(&informacion, sizeof(Informacion), 1, archivo);
431         i = 0;

```

```

434         while (!feof(archivo)) {
435             informaciones[i++] = informacion;
436             fread(&informacion, sizeof(Informacion), 1, archivo);
437         }
438
439         /* Cierra el archivo */
440         fclose(archivo);
441     }
442
443     return informaciones;
444 }
445
446 char existeInformacion(int codigoInformacion, Informacion *informacion)
447 {
448     FILE *archivo;
449     char existe;
450
451     /* Abre el archivo en modo lectura */
452     archivo = fopen("informacion.dat", "rb");
453
454     if (archivo == NULL) { /* Si no se pudo abrir el archivo, el valor de archivo es NULL */
455         existe = 0;
456     } else {
457         existe = 0;
458
459         /* Se busca el registro cuyo código coincida con codigoInformacion */
460         fread(&informacion, sizeof(*informacion), 1, archivo);
461         while (!feof(archivo)) {
462             if ((*informacion).codigo == codigoInformacion) {
463                 existe = 1;
464                 break;
465             }
466             fread(&informacion, sizeof(*informacion), 1, archivo);
467         }
468
469         /* Cierra el archivo */
470         fclose(archivo);
471     }
472 }

```

```

474     return existe;
475 }
476
477 char insertarInformacion(Informacion informacion)
478 {
479     FILE *archivo;
480     char insercion;
481
482     /* Abre el archivo para agregar datos al final */
483     archivo = fopen("Informacion.dat", "ab"); /* Añade datos al final. Si el archivo no existe, es creado */
484
485     if (archivo == NULL) { /* Si no se pudo abrir el archivo, el valor de archivo es NULL */
486         insercion = 0;
487     }
488     else {
489         fwrite(&informacion, sizeof(informacion), 1, archivo);
490         insercion = 1;
491     }
492
493     /* Cierra el archivo */
494     fclose(archivo);
495 }
496
497 return insercion;
498 }
499
500 /* Eliminación lógica de un registro */
501 char eliminarInformacion(int codigoInformacion)
502 {
503     FILE *archivo;
504     FILE *auxiliar;
505     Informacion informacion;
506     char elimina;
507
508     /* Abre el archivo para leer */
509     archivo = fopen("Informacion.dat", "rb"); /* Modo lectura/escritura. Si el archivo no existe, es creado */
510
511     if (archivo == NULL) { /* Si no se pudo abrir el archivo, el valor de archivo es NULL */
512         elimina = 0;
513     }
514     else {
515         while (!feof(archivo)) {
516             if (informacion.codigo != VALOR_CENTINELA) {
517                 fwrite(&informacion, sizeof(informacion), 1, temporal);
518             }
519             fread(&informacion, sizeof(informacion), 1, archivo);
520         }
521         /* Se cierran los archivos antes de borrar y renombrar */
522         fclose(archivo);
523         fclose(temporal);
524
525         remove("Informacion.dat");
526         rename("temporal.dat", "Informacion.dat");
527
528         elimina = 1;
529     }
530
531     return elimina;
532 }
533
534 char modificarInformacion(Informacion informacion)
535 {
536     FILE *archivo;
537     char modifica;
538     Informacion informacion2;
539
540     /* Abre el archivo para lectura/escritura */
541     archivo = fopen("Informacion.dat", "rb+");
542
543     if (archivo == NULL) { /* Si no se pudo abrir el archivo, el valor de archivo es NULL */
544         modifica = 0;
545     }
546     else {
547         modifica = 0;
548         fread(&informacion2, sizeof(informacion2), 1, archivo);
549         while (!feof(archivo)) {
550             if (informacion2.codigo == informacion.codigo) {
551                 fseek(archivo, ftell(archivo) - sizeof(informacion), SEEK_SET);
552                 fwrite(&informacion, sizeof(informacion), 1, archivo);
553             }
554             else {
555                 /* Se busca el registro que se quiere borrar. Cuando se encuentra, se sitúa en esa posición mediante la
556                 función fseek y luego se modifica el campo clave de ese registro mediante algún valor centinela, eso se logra
557                 con fwrite. Hasta allí se ha logrado una eliminación LÓGICA. Porque el registro sigue ocupando espacio en el archivo físico */
558                 elimina = 0;
559                 fread(&informacion, sizeof(informacion), 1, archivo);
560                 while (!feof(archivo)) {
561                     if (informacion.codigo == codigoInformacion) {
562                         fseek(archivo, ftell(archivo) - sizeof(informacion), SEEK_SET);
563                         informacion.codigo = VALOR_CENTINELA;
564                         fwrite(&informacion, sizeof(informacion), 1, archivo);
565                         elimina = 1;
566                         break;
567                     }
568                     fread(&informacion, sizeof(informacion), 1, archivo);
569                 }
570             }
571             /* Cierra el archivo */
572             fclose(archivo);
573         }
574
575         return elimina;
576     }
577 }
578
579 char eliminacionFisica()
580 {
581     FILE *archivo;
582     FILE *temporal;
583     Informacion informacion;
584     char elimina;
585
586     archivo = fopen("Informacion.dat", "rb");
587     temporal = fopen("temporal.dat", "wb");
588
589     if (archivo == NULL || temporal == NULL) {
590         elimina = 0;
591     }
592     else {
593         /* Se copia en el archivo temporal los registros válidos */
594         modifica = 1;
595         break;
596     }
597     fread(&informacion2, sizeof(informacion2), 1, archivo);
598 }
599
600 /* Cierra el archivo */
601 return modifica;
602 }
603
604 char guardarReporte()
605 {
606     FILE *archivo;
607     char guardado;
608     Informacion *informaciones;
609     int numeroInformaciones;
610     int i;
611
612     informaciones = obtenerInformaciones(&numeroInformaciones); /* Retorna un vector dinámico de registros */
613
614     if (numeroInformaciones == 0) {
615         guardado = 0;
616     }
617     else {
618         /* Abre el archivo en modo texto para escritura */
619         archivo = fopen("reporte.txt", "w");
620
621         if (archivo == NULL) { /* Si no se pudo abrir el archivo, el valor de archivo es NULL */
622             guardado = 0;
623         }
624         else {
625             fprintf(archivo, "\n\t\t ==> LISTADO DE PRODUCTOS REGISTRADOS <==\n");
626             fprintf(archivo, "-----\n");
627             fprintf(archivo, "%8s\t%-20s\t%-15s\n", "FECHA O NUMERO", "NOMBRE DEL REGISTRO", "DESCRIPCION", "ARCHIVO");
628             fprintf(archivo, "-----\n");

```

```

630 /* Se recorre el vector dinámico de registros */
631 for (i = 0; i < numeroInformaciones; i++) {
632     if (informaciones[i].codigo != VALOR_CENTINELA) {
633         fprintf(archivo, "%s %s-20.20d%15s%15s\n", informaciones[i].codigo, informaciones[i].nombre, informaciones[i].descrip
634     }
635 }
636
637 guardado = 1;
638
639 /* Cierra el archivo */
640 fclose(archivo);
641 }
642 }
643
644 return guardado;
645 }
646
647 int leerca(char *cad, int n)
648 {
649     int i, c;
650
651     /* Hay que verificar si el buffer está limpio o si hay un '\n'
652     dejado por scanf y, en ese caso, limpiarlo:
653     */
654
655     /* 1 COMPROBACIÓN DE DATOS INICIALES EN EL BUFFER */
656
657     /* Empezamos leyendo el primer caracter que haya en la entrada. Si es
658     EOF, significa que no hay nada por leer, así que cerramos la cadena,
659     dejándola "vacía" y salimos de la función retornando un valor de 0
660     o falso, para indicar que hubo un error */
661     c = getchar();
662     if (c == EOF) {
663         cad[0] = '\0';
664         return 0;
665     }
666
667     /* Si el valor leído es '\n', significa que había un caracter de nueva línea
668     dejado por un scanf o función similar. Simplemente inicializamos i a 0,
669
670     para indicar que los siguientes caracteres que leamos iremos asignando a
671     partir del primer caracter de la cadena. */
672     if (c == '\n') {
673         i = 0;
674     } else {
675         /* Si no había un '\n', significa que el caracter que leímos es el primer
676         caracter de la cadena introducida. En este caso, lo guardamos en la
677         posición 0 de cad, e inicializamos i a 1, porque en este caso, como ya
678         tenemos el primer caracter de la cadena, continuaremos agregando
679         caracteres a partir del segundo.
680
681         */
682         cad[0] = c;
683         i = 1;
684     }
685
686     /* 2. LECTURA DE LA CADENA */
687
688     /* El for empieza con un ; porque estamos omitiendo la inicialización del contador,
689     ya que fue inicializado en el punto anterior.
690     Este código lee un caracter a la vez, lo agrega a cad, y se repite hasta que
691     se encuentre un fin de línea, fin de archivo, o haya leído la cantidad máxima
692     de caracteres que se le indicó. Luego, cierra la cadena agregando un '\0'
693     al final. Todo esto es muy similar a la forma en que los compiladores suelen
694     implementar la función fgets, sólo que en lugar de getchar usan getc o fgetc
695     */
696     for (; i < n - 1 && (c = getchar()) != EOF && c != '\n'; i++) {
697         cad[i] = c;
698     }
699     cad[i] = '\0';
700
701     /*3. LIMPIEZA DEL BUFFER */
702
703     /* Finalmente limpiamos el buffer si es necesario */
704     if (c != '\n' && c != EOF) /* es un caracter */
705         while ((c = getchar()) != '\n' && c != EOF);
706
707     /* La variable c contiene el último caracter leído. Recordemos que había 3 formas
708     de salir del for: que hayamos encontrado un '\n', un EOF, o que hayamos llegado


```

```

709     de salir del for: que hayamos encontrado un '\n', un EOF, o que hayamos llegado
710     al máximo de caracteres que debemos leer. Si se da cualquiera de los dos
711     primeros casos, significa que leímos todo lo que había en el buffer, por lo que
712     no hay nada que limpiar. En el tercer caso, el usuario escribió más caracteres
713     de los deseados, que aún están en el buffer, por lo que hay que quitarlos, para
714     lo cual usamos el método que vimos poco más arriba
715     */
716     return 1;
717 }
718
719 void tituloPrincipal()
720 {
721     int i;
722     printf("\n =====\n");
723     printf("\t\t\t\t\t PROYECTO FINAL\n");
724     printf("\t\t\t\t\t Base de datos: Creación, reportes, eliminación, búsqueda y actualización\n");
725     printf("\t\t\t\t\t Arroyo Chavarria Jose Luis\n");
726     printf("\n =====\n");
727
728     i = 0;
729     putchar('\n');
730     for (; i < 80; i++) {
731         putchar('_');
732     }
733 }

```

Resultados:

 C:\Users\luisw\Desktop\tareas UNAM\segundo semestre\datos y estructuras 1\Proyecto final\Proyecto final .exe

```

=====
PROYECTO FINAL
Base de datos: Creación, reportes, eliminación, búsqueda y actualización
Arroyo Chavarria Jose Luis
=====

MENU PRINCIPAL

[1]. Insertar nuevo registro
[2]. Mostrar listado de registros
[3]. Eliminar un registro
[4]. Buscar registro por numero o fecha
[5]. Modificar un registro
[6]. Eliminación física de registros
[7]. Salir

Ingrese su opción: [1]

```

C:\Users\luisw\Desktop\tareas UNAM\segundo semestre\datos y estructuras 1\Proyecto final\Proyecto final .exe

```
=====
                        PROYECTO FINAL
Base de datos: Creación, reportes, eliminación, búsqueda y actualización
Arroyo Chavarria Jose Luis
=====

==> INSERTAR INFORMACION <==

Codigo: 652020

Fecha o numero: 06/05/2020
Nombre del registro: Registro de Prueba
Descripcion: Ejemplo
Archivo: Prueba.pdf

El registro fue insertado correctamente

Desea seguir ingresando registros? [S/N]: N
```

C:\Users\luisw\Desktop\tareas UNAM\segundo semestre\datos y estructuras 1\Proyecto final\Proyecto final .exe

```
=====
                        PROYECTO FINAL
Base de datos: Creación, reportes, eliminación, búsqueda y actualización
Arroyo Chavarria Jose Luis
=====

==> LISTADO DE REGISTROS INGRESADOS <==

-----
CODIGO      FECHA O NUMERO      REGISTRO      DESCRIPCION      ARCHIVO
-----
652020      06/05/2020          Registro de Prueba  Ejemplo          Prueba.pdf

Desea guardar el reporte en un archivo de texto? [S/N]: S

El reporte fue guardado con éxito
```

reporte.txt: Bloc de notas

Archivo Edición Formato Ver Ayuda

```
==> LISTADO DE DATOS REGISTRADOS <==

-----
CODIGO      FECHA O NUMERO      REGISTRO      DESCRIPCION      ARCHIVO
-----
652020      06/05/2020          Registro de Prueba  Ejemplo          Prueba.pdf
```

C:\Users\luisw\Desktop\tareas UNAM\segundo semestre\datos y estructuras 1\Proyecto final\Proyecto final .exe

```
=====
                        PROYECTO FINAL
Base de datos: Creación, reportes, eliminación, búsqueda y actualización
Arroyo Chavarria Jose Luis
=====

==> BUSCAR REGISTRO POR CODIGO <==

Codigo del registro: 652020

Codigo del registro: 652020

Fecha o numero del registro: 06/05/2020
Nombre del registro: Registro de Prueba
Descripcion del registro: Ejemplo
Archivo del registro: Prueba.pdf

Desea seguir buscando algún producto? [S/N]:
```

C:\Users\luisw\Desktop\tareas UNAM\segundo semestre\datos y estructuras 1\Proyecto final\Proyecto final.exe

```
=====
                          PROYECTO FINAL
Base de datos: Creación, reportes, eliminación, búsqueda y actualización
Arroyo Chavarria Jose Luis
=====

==> MODIFICAR REGISTRO POR CODIGO <==

Codigo del registro: 652020

Fecha o numero del registro: 06/05/2020

Nombre del registro: Registro de Prueba
Descripcion: Ejemplo
Archivo: Prueba.pdf

Elija los datos a modificar

Fecha o numero: 06/05/2020
Desea modificar la fecha o numero? [S/N]: N

Nombre del registro actual: Registro de Prueba
Desea modificar el nombre del registro? [S/N]: S
Nuevo nombre del registro: Hola

Descripcion: Ejemplo
Desea modificar la descripcion? [S/N]: s
Nueva descripcion del registro: Adios

Archivo del registro actual: Prueba.pdf
Desea modificar el archivo del registro? [S/N]: n

Está seguro que desea modificar los datos del registro? [S/N]: s

El registro fue modificado correctamente

Desea modificar algún otro registro? [S/N]: _
```

C:\Users\luisw\Desktop\tareas UNAM\segundo semestre\datos y estructuras 1\Proyecto final\Proyecto final.exe

```
=====
                          PROYECTO FINAL
Base de datos: Creación, reportes, eliminación, búsqueda y actualización
Arroyo Chavarria Jose Luis
=====

==> LISTADO DE REGISTROS INGRESADOS <==

-----
CODIGO      FECHA O NUMERO      REGISTRO      DESCRIPCION      ARCHIVO
-----
652020      06/05/2020          Hola          Adios             Prueba.pdf

Desea guardar el reporte en un archivo de texto? [S/N]: _
```

C:\Users\luisw\Desktop\tareas UNAM\segundo semestre\datos y estructuras 1\Proyecto final\Proyecto final.exe

```
=====
                          PROYECTO FINAL
Base de datos: Creación, reportes, eliminación, búsqueda y actualización
Arroyo Chavarria Jose Luis
=====

==> ELIMINAR REGISTRO POR FECHA O NUMERO <==

Codigo del registro: 652020
Codigo del registro: 652020

Fecha o numero del registro: 06/05/2020
Nombre del registro: Hola
Descripcion del registro: Adios
Archivo del registro: Prueba.pdf

Seguro que desea eliminar el registro? [S/N]: S

Registro eliminado satisfactoriamente.

Desea eliminar otro registro? [S/N]: N
```

C:\Users\luisw\Desktop\tareas UNAM\segundo semestre\datos y estructuras 1\Proyecto final\Proyecto final .exe

```
=====
PROYECTO FINAL
Base de datos: Creación, reportes, eliminación, búsqueda y actualización
Arroyo Chavarria Jose Luis
=====

==> LISTADO DE REGISTROS INGRESADOS <==
-----
CODIGO      FECHA O NUMERO      REGISTRO      DESCRIPCION      ARCHIVO
-----
Desea guardar el reporte en un archivo de texto? [S/N]: _
```

C:\Users\luisw\Desktop\tareas UNAM\segundo semestre\datos y estructuras 1\Proyecto final\Proyecto final .exe

```
=====
PROYECTO FINAL
Base de datos: Creación, reportes, eliminación, búsqueda y actualización
Arroyo Chavarria Jose Luis
=====

==> ELIMINAR FÍSICAMENTE REGISTROS DEL ARCHIVO <==

Seguro que desea proceder con la eliminación física? [S/N]: S

La eliminación física se realizó con éxito.
```

Conclusiones:

Referencias:

- [https://es.wikipedia.org/wiki/Memoria_dinámica_\(programación\)](https://es.wikipedia.org/wiki/Memoria_dinámica_(programación))
- [https://www.ecured.cu/Pila_\(Estructura_de_datos\)](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](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