



# INSTITUTO POLITÉCNICO NACIONAL

## ESCUELA SUPERIOR DE CÓMPUTO

### ALGORITMOS Y ESTRUCTURA DE DATOS

GRUPO: 2CV11

ALUMNO:

JONATHAN SAID GÓMEZ MARBÁN

PRÁCTICA 1:

“Manejo de cadenas”

**Introducción**

En esta práctica desarrollaremos algunos ejercicios que nos permitan, leer e imprimir cadenas, así como invertir la cadena o imprimirlo n número de veces. Para ello, utilizaremos temas tales como, memoria dinámica, es decir, se usa las funciones malloc, free, calloc y realloc, también se tiene que hacer uso de apuntadores. En la práctica solo se ocuparon las primeras dos funciones mencionadas.

Recordando un poco de esto, la función malloc nos permite reservar memoria en un numero específico de bytes y nos regresa un apuntador a void, el cual, con hacer un cast, se puede hacer apuntador a el tipo que se requiera. La función free, nos permite liberar espacio de la memoria, ya que la función malloc solo nos permite reservarla, sin embargo, no la libera, para eso es que se usa la función free. La función calloc es similar a malloc, de hecho, ambas nos permiten reservar espacios de memoria, la diferencia es que calloc inicializa todos los bits a cero. Mientras que la función realloc nos permite cambiar el tamaño de memoria asignada previamente, es decir, que la memoria que asignamos fue demasiada o insuficiente y requerimos ajustar su tamaño. Todas estas funciones están implementadas en la librería stdlib.h por lo que hay que incluirla dentro de nuestro programa.

En la librería stdlib.h también encontramos otra función que es de utilidad para el manejo de cadenas y que se requiere usar en la práctica, la cual es la función strlen que nos permite saber el total (en entero) de caracteres que forman una cadena, la función cuenta todos los caracteres hasta llegar a la posición del apuntador a carácter en cuestión en donde es igual a NULL, por ejemplo, la cadena de la que se quiere saber el número de caracteres que contiene es “hola”, pasando como parámetro el apuntador donde esta guardada esta cadena a la función strlen, regresara el valor de 4.

Otra función importante es sizeof, que regresa el tamaño en bytes de un tipo de dato o de alguna variable, por tanto, como parámetro recibe un tipo de dato o una variable. Por ejemplo, sizeof (int) regresa el número 4.

Estas 2 ultimas funciones descritas, nos ayudaran a determinar el tamaño de memoria que queremos reservar con la función malloc, explicada con anterioridad, ya que podremos medir cuantos caracteres tiene una cadena y cuanto equivale cada carácter.

Es importante también mencionar el tema de arreglos, para esta práctica utilizamos más en específico lo que se conoce como vector, recordemos que los arreglos también son apuntadores y tienen memoria estática. A través de estos dos temas (apuntadores y memora dinámica), desarrollaremos y resolveremos los problemas propuestos en la práctica.

**Problemas relacionados al tema a los cuales te enfrentaste al programar la práctica.**

Uno de los principales problemas que tuve fue el desbordamiento de la memoria o que la cadena que introducía no salía completa en la impresión, ya que, para leer la cadena, yo utilice la función fgets, porque si usaba scanf que es la más usual o común, la cadena la cortaba donde encontraba un espacio o el carácter " ", por eso usé la función fgets que es en específico para leer cadenas y la lee toda, sin cortarla al encontrar algún espacio. Pero a fgets (que recibe un apuntador a carácter y un número, el cual es el número máximo de caracteres a leer), se detiene cuando se leen (número máximo -1) caracteres, se lee el carácter de nueva línea (enter) o se llega al final del archivo, lo que ocurra primero. Esto me generó conflictos al momento de leer la cadena, ya que, en un principio, no leyó toda la cadena o leía un carácter de más, debido al número máximo de caracteres que yo introducía.  Entonces al momento que se ejecutaba el programa no imprimía la cadena deseada.

Otro problema que tuve dentro del desarrollo de la practica fue en la función de invertir la cadena, al usar strlen, ya que, por ejemplo, si la cadena que quiero medir es "hola", regresa 5, porque regresa el número total de caracteres incluyendo el salto de línea. Y el otro problema fue hallar en sí la solución a invertir la cadena, lo que yo hice, que me resulto de gran utilidad, fue manejar dentro de un mismo ciclo for dos variables, una que tuviera un paso 1 a 1 positivo, mientras que la otra variable tuviera un paso 1 a 1 negativo (en retroceso), así el contenido de la última posición se pasa a la primera, de la penúltima a la segunda, etc.

En general, esos fueron mis 3 problemas principales que tuve en el desarrollo de la práctica.

**Documentación programa (capturas de pantalla).**

```
#include <stdio.h> /*fgets, sscanf */
#include <stdlib.h> /*malloc, free*/
#include <string.h> /*strlen*/
#include <stdbool.h> /*Para poder declarar variables de tipo booleano
                       y asignales un valor de verdadero o falso*/
```

Librerías usadas. Imagen 1.

Como logramos observar en la imagen 1, lo primero que se hace en el programa es la inclusión de ficheros, más en específico la directiva #include con la cual incluimos el contenido del fichero de cabecera que contiene los prototipos de las funciones de entrada/salida de la biblioteca estándar de C, en los cuales como se logra apreciar en los comentarios del programa, la función fgets y sscanf que explicaremos más adelante.

Sabemos también que es en esta biblioteca estándar donde también se encuentran la función que es muy usado como printf. Después se incluye stdlib.h, en donde nuevamente es importante destacar que en los comentarios se pone la función malloc y free que son usadas en el programa. Posteriormente string.h, que de la cual solo ocupamos una función, de este archivo de cabecera, la cual es strlen. Y por último stdbool.h que permite usar bool como un tipo de datos booleano.

```
/*Función imprimirMenu, no regresa nada, no recibe parámetros.
Esta función simplemente imprime en pantalla el menú que se
especifica en la práctica.*/
void imprimirMenu()
{
    printf("\n\t\t:::Menu:::");
    printf("\n1. Introducir una nueva cadena");
    printf("\n2. Imprimir la cadena");
    printf("\n3. Invertir la cadena");
    printf("\n4. Concatenar consigo misma");
    printf("\n5. Salir");
}
```

Función imprimirMenu. Imagen 2.

Como se logra apreciar en la imagen 2, creamos una función que cada que la invoquemos imprimirá el menú que se muestra en la imagen.

```
/*introducirNuevaCadena, no regresa nada, recibe como parametro un apuntador a caracter.
Pide al usuario introducir una cadena y luego lee esa cadena introducida*/
void introducirNuevaCadena(char *vectorC)
{
    printf("Introduce una cadena: ");
    fgets(vectorC, 256, stdin);
}
```

Función introducirNuevaCadena. Imagen 3.

Le permite al usuario introducir una cadena, que será leída y almacenada. Note que se usa fgets y este recibe un apuntador a carácter, lee 255, porque recordando que lee el número máximo de caracteres – 1 y como ultimo parámetro el flujo que es un apuntador a un objeto FILE que identifica un flujo de entrada; stdin se puede utilizar como argumento para leer desde la entrada estándar.

```c
/*imprimirCadena, no regresa nada, recibe como parametro un apuntador a caracter.
Hace la impresion en la consola o pantalla de una cadena*/
void imprimirCadena(char *vectorC)
{
    printf("%s", vectorC);
}
```

Función imprimirCadena. Imagen 4.

Simplemente la función recibe un apuntador a cadena y con uso de printf, imprimimos la cadena recibida.

```c
/*inicializarNuevaCadena regresa un apuntador a caracter, recibe como parametro un apuntador a caracter.
Recibe un apuntador a caracter, despues reserva espacio en la memoria a traves de la funcion malloc donde
multiplica el tamaño del apuntador por el tamaño de un caracter, lo asigna a una variable y m regresa dicha
variable.*/
char *inicializarNuevaCadena(char *vectorC)
{
    char *cadena;
    cadena = (char *)malloc(sizeof(char) * (strlen(vectorC)));
    return cadena;
}
```

Función introducirNuevaCadena. Imagen 5.

Con esta función asignamos a una variable tipo apuntador a caracter que creamos dentro de la función, luego le asignamos el espacio de memoria que reservaremos y regresamos esa variable. Como notamos, en los comentarios se describe mas a detalle que se realiza. También observamos que se hace un cast al retorno de la función malloc, lo estamos convirtiendo en un apuntador a carácter.

```c
/*invertirCadena, recibe un apuntador a caracter, regresa un apuntador a caracter
Esta funcion nos permite cambiar la cadena, haciendo que quede de manera opuesta,
es decir, invertir la cadena; primero reservamos espacio en la memoria y lo almacenamos
en un apuntador a caracter, luego, a traves de recorrer las posiciones del
parametro y las de otro apuntador a caracter y asignando valores, se logra  invertir la cadena,
finaliza el ciclo e imprime la cadena invertida y la funcion me regresa esa cadena*/
char *invertirCadena(char *vectorC)
{
    char *cadena;
    cadena = inicializarNuevaCadena(vectorC);
    for (int i = strlen(vectorC) - 2, k = 0; i >= 0; i--, k++)
    {
        cadena[k] = vectorC[i];
    }
    cadena[strlen(vectorC) - 1] = '\0';
    imprimirCadena(cadena);
    return cadena;
}
```

Función invertirCadena. Imagen 6.

En la imagen 6, en los comentarios ya se ha explicado gran parte de lo que realiza la función, me gustaría mas enfocarme en el for que se observa en dicha imagen.

La variable i se inicializa con el tamaño del apuntador menos 2, esto es porque strlen(vectorC) regresa el número total de caracteres incluyendo el salto de línea, por ejemplo, si la cadena a medir fuera “hola” regresa 5, al restarle 2, se obtiene 3, entonces i va desde 3 hasta 0, mientras que k se inicializa en 0 y tiene n paso 1 a 1 positivo. Es por ello que i se inicializa en el tamaño del parámetro recibido menos 2 y k siempre empieza en 0. Después, asignando los valores de el parámetro recibido en la posición de i (del fin al inicio de la cadena) al apuntador creado dentro de la función (que va de inicio a fin, que es el valor que tiene k). Así obtenemos la cadena invertida, luego nos aseguramos de que en la posición final tengamos el caracter de fin de cadena. Imprimimos la cadena con la función imprimirCadena y regresamos la cadena invertida.

```c
/*concatenarCadena, recibe un entero y un apuntador a carcter y regresa un apuntador a caracter
Primero se lee el numero de veces que se va a repetir la cadena, luego a traves de if's, analizamos
en que diferente caso es en el que se encuentra, ejecuta el codigo correspondiente, por ultimo
imprime la cadena y la funcion me regresa dicha cadena.*/
char *concatenarCadena(int numeroVeces, char *vectorC)
{
    printf("\nNumero de veces: ");
    scanf("%d", &numeroVeces);
    if (numeroVeces == 0)
    {
        printf("");
        return 0;
    }
    bool eraPositivo = true;
    if (numeroVeces < 0)
    {
        eraPositivo = false;
        numeroVeces *= -1;
    }
    char *cadena;
    cadena = (char *)malloc(sizeof(char) * (strlen(vectorC) - 1) * numeroVeces + sizeof(char));
    for (int i = 0; i < numeroVeces; i++)
    {
        if (eraPositivo)
        {
            for (int j = 0; j < strlen(vectorC) - 1; j++)
            {
                cadena[j + ((strlen(vectorC) - 1) * i)] = vectorC[j];
            }
        }
        else
        {
            for (int j = strlen(vectorC) - 2, k = 0; j >= 0; j--, k++)
            {
                cadena[k + ((strlen(vectorC) - 1) * i)] = vectorC[j];
            }
        }
    }
    cadena[(strlen(vectorC) - 1) * numeroVeces] = '\0';
    imprimirCadena(cadena);
    return cadena;
}
```

Función cocatenarCadena. Imagen 7.

En los comentarios de la función, se logra observar que se ha rescrito un poco la misma. Me concentrare en los distintos casos que pueden existir, el primero de ellos es que el numero de veces sea igual a 0, con lo cual nosotros imprimiremos el carácter " " o vacío. Y salimos de la función en ese mismo momento, mientras que, si no es 0, creamos una variable de tipo bool que me ayudara a saber si es negativo o positivo, dicha variable se inicializa en true, si el valor del numero de veces es negativo, entonces cambiamos el valor a false y multiplicamos el numero de veces por -1, para hacerlo positivo. Creamos un apuntador a carácter y le asignamos el espacio de memoria. Luego hay un for que va desde 0 hasta mi

número de veces (este for hará que se imprima las veces que se indicó). Tenemos 2 casos, el primero es que el numero de veces haya sido positivo, por lo cual el valor de la variable tipo bool es verdadero, con otro for que empiece en 0 y termine en el tamaño del parámetro (strlen(vectorC)-1).

Observamos que se asigna el parámetro en la posición j (que empieza en 0) sobre el apuntador creado dentro de la función, y se asigna en la posición
j + ((strlen(vectorC) - 1) * i)

Para la primera vez que se ejecutan ambos for, i y j son 0, si se hace la matemática la posición de la variable cadena será 0, para la segunda vez que se ejecuta el for anidado j=1, pero i=0. Si se hace la matemática, siempre se multiplica por 0 entonces cadena valdrá el valor de j cuando i valga 0, este for interno se repite hasta que llegue a el tamaño del parámetro, una vez termina este, i avanza 1, es decir i=1, cuando i tiene este valor y j=0 entonces iniciamos justamente en el siguiente valor del ultimo en el que fue escrito, así evitamos sobre escribir en las mismas posiciones y de esta forma se concatena la cadena el n numero de veces que se requiere.

Ahora, en el caso de que número de veces sea negativo, se tiene la misma lógica para no sobre escribir en la misma posición, pero también se usan dos variables dentro del for para ir invirtiéndola al mismo tiempo (como se explicó en la función de invertirCadena). Luego nos aseguramos de que en la posición final tengamos el caracter de fin de cadena. Imprimimos la cadena con la función imprimirCadena y regresamos la cadena.

```c
int main()
{
    char valorUsuario[255];
    int seleccionNum;
    char seleccion;
    char *nuevaCadena;

    introducirNuevaCadena(valorUsuario);

    do
    {
        imprimirMenu();

        printf("\nOpcion seleccionada: ");
        fgets(&seleccion, 3, stdin);
        sscanf(&seleccion, "%d", &seleccionNum);

        switch (seleccionNum)
        {
        case 1:
            printf("\nTu seleccion fue introducir una nueva cadena: \n");
            introducirNuevaCadena(valorUsuario);
            break;
        case 2:
            printf("\nTu seleccion fue imprimir la cadena: \n");
            imprimirCadena(valorUsuario);
            break;
        case 3:
            printf("\nTu seleccion fue invertir cadena: \n");
            nuevaCadena = invertirCadena(valorUsuario);
            free(nuevaCadena);
            break;
        case 4:
            printf("\nTu seleccion fue concatenar cadena");
            int numeroVeces;
            nuevaCadena = concatenarCadena(numeroVeces, valorUsuario);
            free(nuevaCadena);
            break;
        case 5:
            printf("\nTu seleccion fue salir \nAdios.");
            return 0;
        default:
            printf("\nPon atencion");
        }
    } while (true);
}
```

Función main. Imagen 8.

En el main es donde utilizamos todas las funciones descritas con anterioridad, y solo se han declarado algunas nuevas variables que utilizaremos para pasar como parámetros, un do-while en el que esta adentro la mayoría de las funciones contenidas en casos de un switch.

Nótese que es aquí donde utilizamos la función free, para liberar el espacio de la memoria.

```
fgets(&seleccion, 3, stdin);
sscanf(&seleccion, "%d", &seleccionNum);
```

Leer la selección. 8.1.

Primero con fgets se obtiene la cadena, se usan 3 para solo leer dos caracteres y lo pones en selección (es un solo carácter, es decir, solo se guarda un valor), luego con sscanf formatea esa cadena hacia un entero, y lo guarda en seleccionNum.

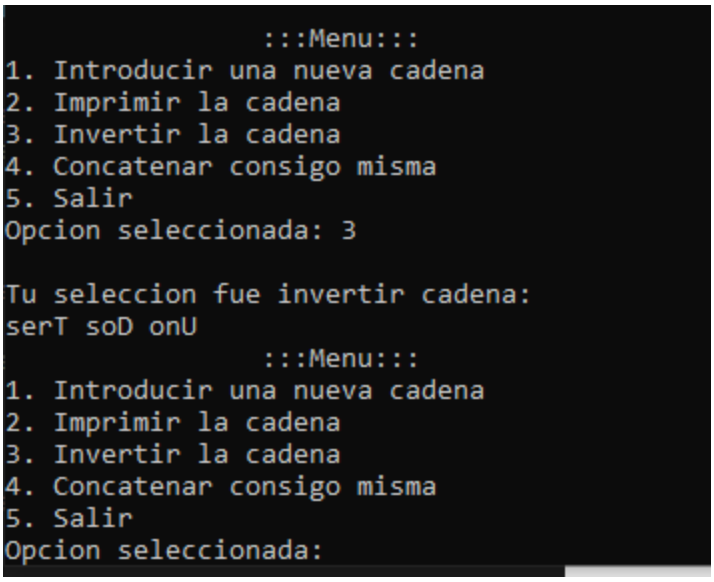
A continuación, se muestran las capturas del programa ejecutándose desde el cmd los valores de entrada proporcionados en la práctica.

Para el valor de entrada "Uno Dos Tres":

```
C:\Users\jona-\Documents\Jona\ESCOM\semestre 2\Estructura de datos>gcc manejo_de_cadenas.c

C:\Users\jona-\Documents\Jona\ESCOM\semestre 2\Estructura de datos>a
Introduce una cadena: Uno Dos Tres

               :::Menu:::
1. Introducir una nueva cadena
2. Imprimir la cadena
3. Invertir la cadena
4. Concatenar consigo misma
5. Salir
Opcion seleccionada: 2

Tu seleccion fue imprimir la cadena:
Uno Dos Tres

               :::Menu:::
1. Introducir una nueva cadena
2. Imprimir la cadena
3. Invertir la cadena
4. Concatenar consigo misma
5. Salir
Opcion seleccionada:
```

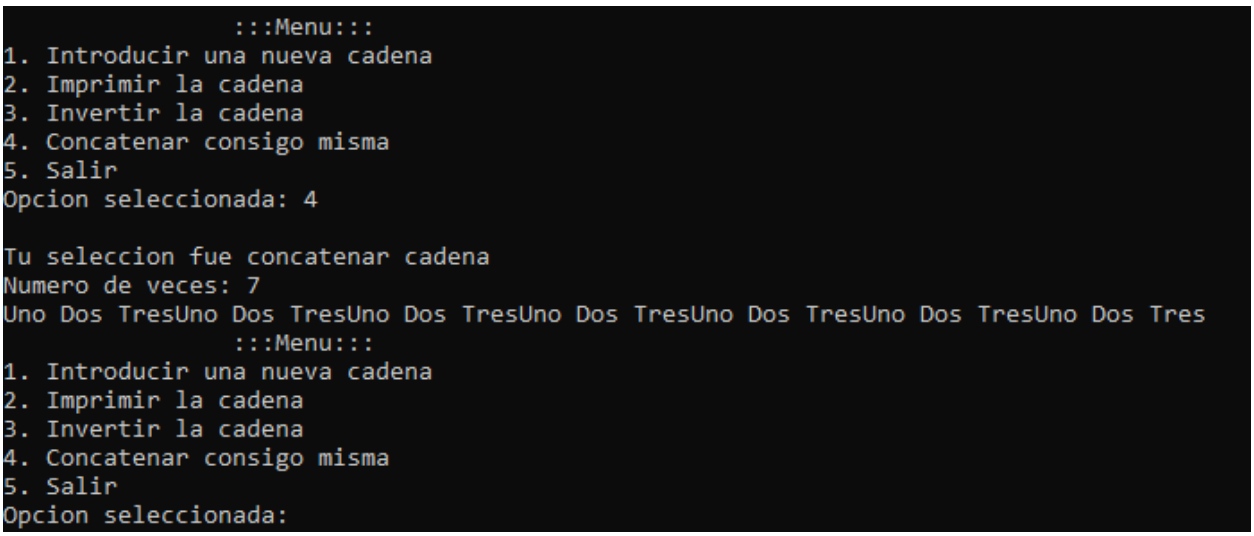
Introducción de la cadena, elección 2. Imagen 9.

Como logramos observar en imagen 9, lo primero que hace el programa es solicitar una cadena, luego imprime el menú con las 5 opciones, después requiere que ingreses la opción que deseas realizar. En este caso se introduce 2, y se imprime la cadena. Posteriormente se vuelve a imprimir el menú.



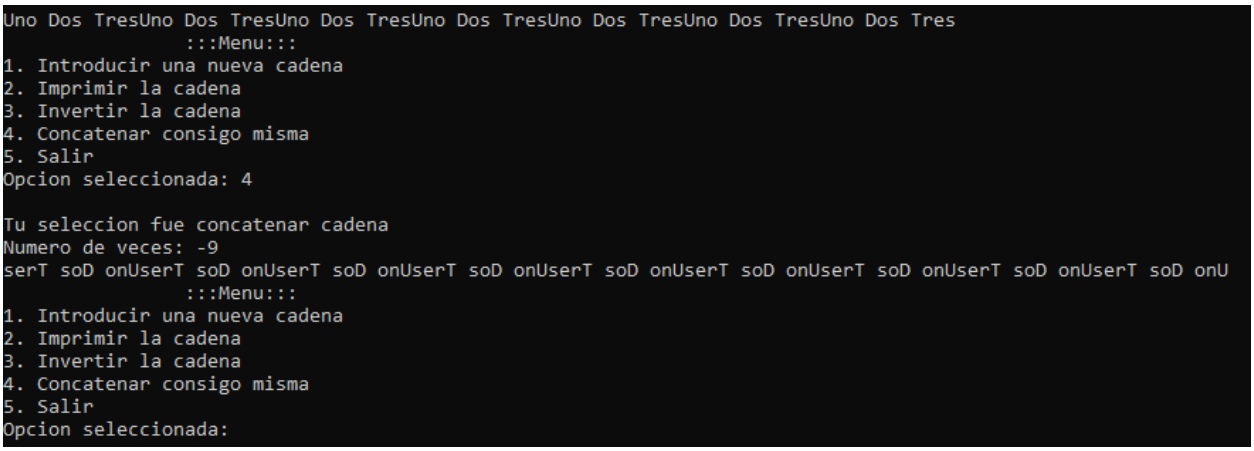

Invertir cadena ejemplo con cadena "Uno Dos Tres". Imagen 10.

En la imagen 10 (que sigue el mismo programa en ejecución que el que se muestra en la imagen 9), se muestra que hemos introducido el numero 3, es decir, invertir la cadena, luego, nos imprime esa cadena invertida y vuelve a imprimir el menú.



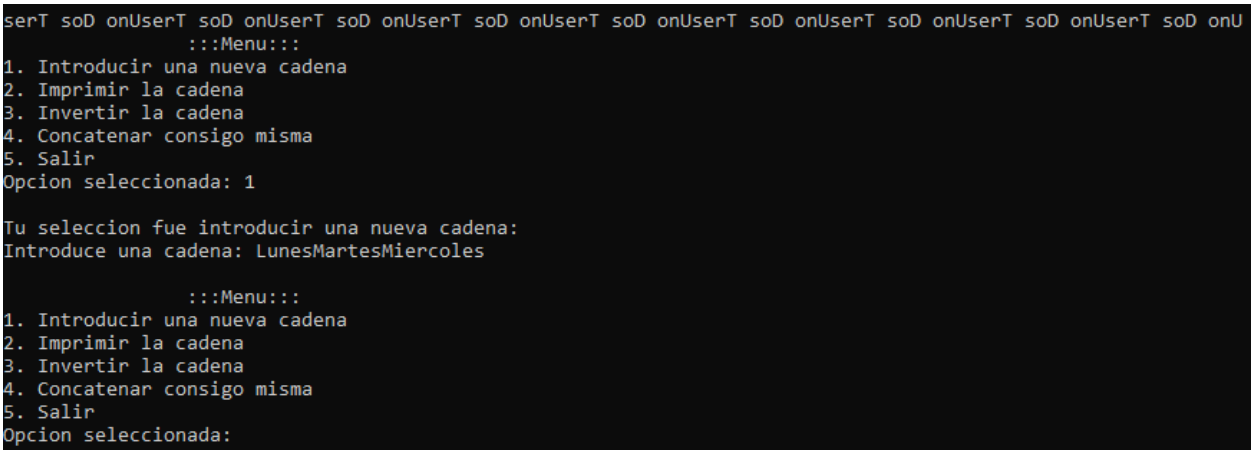

Concatenar cadena "Uno Dos Tres" 7 veces. Imagen 11.

Como observamos en esta ocasión elegimos la opción 4 y le hemos indicado que concatene 7 veces la cadena consigo misma.




Invertir y concatenar cadena "Uno Dos Tres" 7 veces. Imagen 11.

En la imagen 11 se nos muestra como se vuelve a elegir la opción 4, pero esta vez primero se invierte la cadena por ser un numero negativo y luego se concatena  veces.




Introducir nueva cadena. Imagen 12

En la imagen 12 se muestra como hemos elegido la opción 1, que es introducir una nueva cadena. Y hemos escrito la cadena "LunesMartesMiercoles".

```
                ::::Menu:::
1. Introducir una nueva cadena
2. Imprimir la cadena
3. Invertir la cadena
4. Concatenar consigo misma
5. Salir
Opcion seleccionada: 2

Tu seleccion fue imprimir la cadena:
LunesMartesMiercoles

                ::::Menu:::
1. Introducir una nueva cadena
2. Imprimir la cadena
3. Invertir la cadena
4. Concatenar consigo misma
5. Salir
Opcion seleccionada:
```

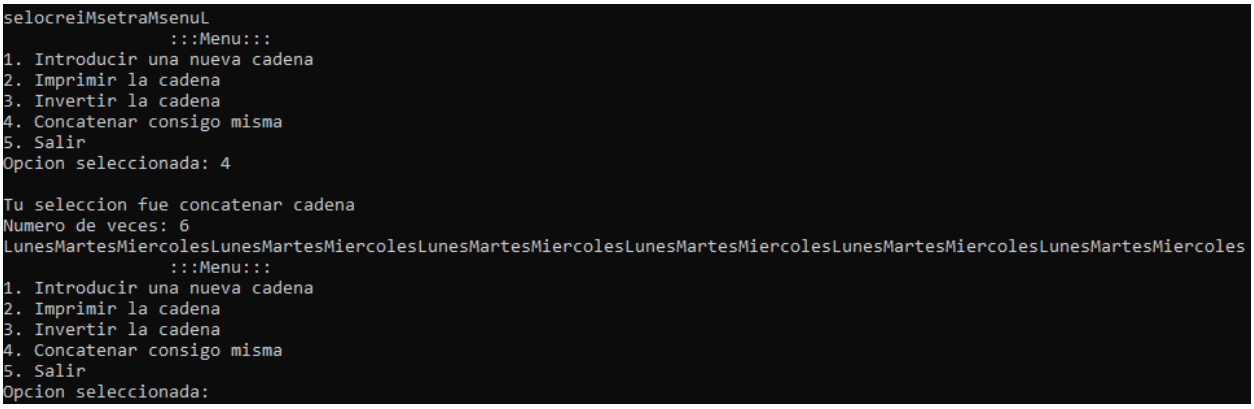
Imprimir la nueva cadena. Imagen 13.

Seleccionamos la opción 2 y nos ha impreso la cadena “LunesMartesMiercoles”. No olvidemos que esa es la nueva cadena introducida.

```
                ::::Menu:::
1. Introducir una nueva cadena
2. Imprimir la cadena
3. Invertir la cadena
4. Concatenar consigo misma
5. Salir
Opcion seleccionada: 3

Tu seleccion fue invertir cadena:
selocreiMsetraMsenuL
                ::::Menu:::
1. Introducir una nueva cadena
2. Imprimir la cadena
3. Invertir la cadena
4. Concatenar consigo misma
5. Salir
Opcion seleccionada:
```

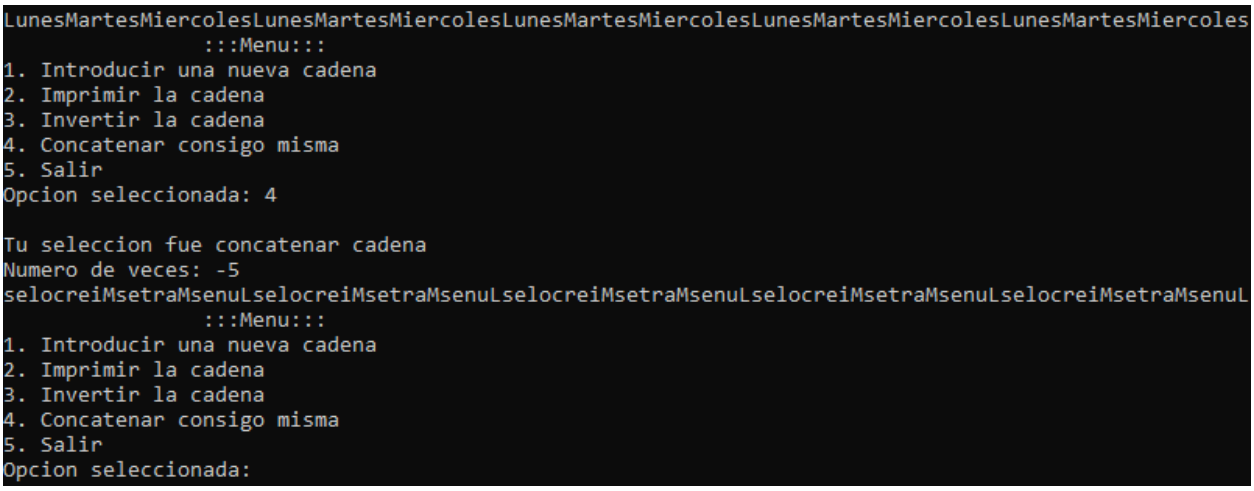
Invertir nueva cadena. Imagen 14.

Observamos que elegimos la opción 3, con esto, se ejecuta el caso 3 del switch de la main, que invoca a la función invertir cadena.




Selección de la opción 4 para la prueba número 2. Imagen 15.

Ahora, hemos seleccionado la opción 4, que me concatena n numero de veces la cadena que haya introducido. En este caso, como se logra apreciar, se ha dicho que se concatene 6 veces.




Invertir segunda cadena propuesta. Imagen 16.

Hemos elegido la opción 4, y queremos que se invierta y después se concatene 5 veces, tal y como se muestra en la imagen.

Ahora, pasaremos a introducir la ultima cadena que fue propuesta:

```
Tu seleccion fue introducir una nueva cadena:
Introduce una cadena: Enero Febrero Marzo Abril Mayo Junio Agosto

               :::Menu:::
1. Introducir una nueva cadena
2. Imprimir la cadena
3. Invertir la cadena
4. Concatenar consigo misma
5. Salir
Opcion seleccionada:
```

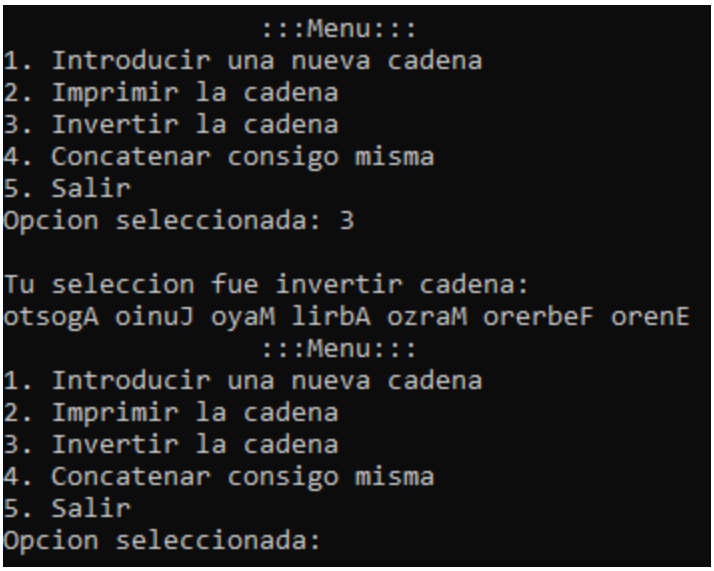
Leer la ultima cadena propuesta. Imagen 17.

En la imagen 17 se observa cómo hemos introducido la cadena “Enero Febrero Marzo Abril Mayo Junio Agosto”, ahora trabajaremos con ella:

```
               :::Menu:::
1. Introducir una nueva cadena
2. Imprimir la cadena
3. Invertir la cadena
4. Concatenar consigo misma
5. Salir
Opcion seleccionada: 2

Tu seleccion fue imprimir la cadena:
Enero Febrero Marzo Abril Mayo Junio Agosto

               :::Menu:::
1. Introducir una nueva cadena
2. Imprimir la cadena
3. Invertir la cadena
4. Concatenar consigo misma
5. Salir
Opcion seleccionada:
```

Impresión de la última cadena propuesta. Imagen 18.

Logramos notar en la imagen 18, que se imprimió correctamente la cadena, ahora la invertiremos:



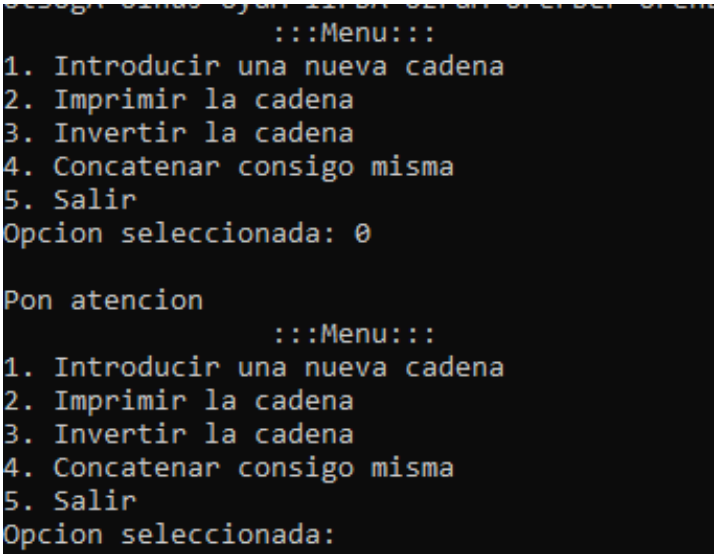

Invertir cadena "Enero Febrero Marzo Abril Mayo Junio Agosto". Imagen 19.

Se muestra como se ha invertido la cadena de manera correcta, ahora mostraremos como es que la concatenamos 5 veces y como la invertimos y concatenamos 5 veces igualmente.


Concatenar 5 veces tanto normal como invertida la ultima cadena propuesta. Imagen 20.

Para apreciar mejor la imagen, se recomienda hacer un zoom al documento, con ello se podrá lograr ver bien la imagen, la imagen 20 no la puedo hacer mas grande, ni la puedo recortar en varias, es por ello que se ve así.

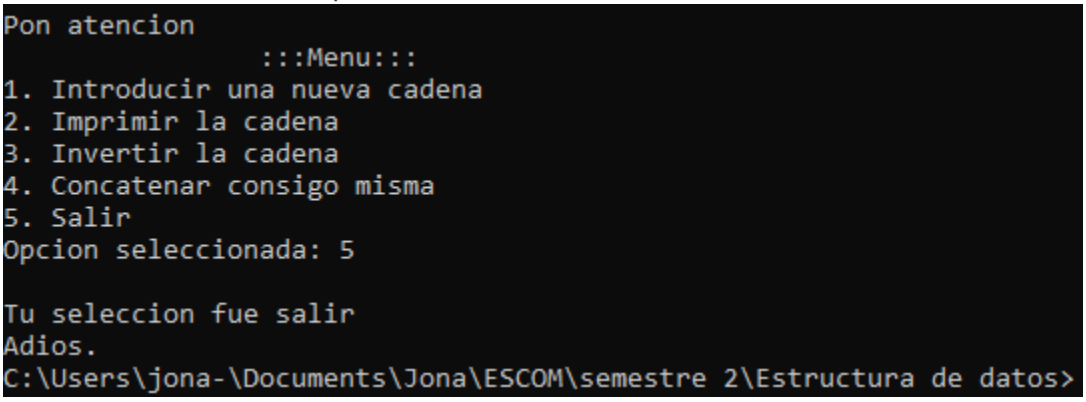
Ahora solo falta ver, que es lo que pasa cuando introduzco un valor diferente de 1 a 5.




Caso default. Imagen 21.

En la imagen 8 (función main) logramos apreciar como el caso default de switch, es simplemente que imprima en pantalla "Pon atencion", pero eso no hace que se salga del programa, sino que simplemente vuelve a imprimir el menú esperando a que una opción valida sea introducida.

Por último, veremos la opción 5, la cual es salir.




Opción 5. Imagen 22.

Nos percatamos de que la opción 5 nos imprime "Tu selección fue salir", un salto de línea y "Adios". Con esto, se da por terminado las pruebas del programa.

## Conclusión

En esta práctica me ayudo a entender como funciona y a usar las funciones fgets, sscanf y strlen, ya que, en un principio no sabía como utilizarlas, pero una vez investigadas y practicando con ellas, entendí mejor como usarlas. También entendí mejor la asignación de memoria utilizando la función malloc, pero ahora utilizando las funciones sizeof y strlen para saber el tamaño que requiero asignar a un apuntador.

También aprendí a ocupar 2 variables dentro de un mismo for, ya que, antes de esta solución, no se me ocurría nada para resolver el problema de invertir la cadena.

Aprendí como verificar o como saber cual es el fin de una cadena, o mas bien, como es que una función puede identificar el fin de la cadena y es con los caracteres "\0".

Aunque no lo ocupe dentro de la práctica, cuando investigue las funciones malloc y free, encontré también información acerca de calloc y realloc, y entendí la diferencia que hay entre malloc y realloc.

En sí creo que, el pensar y entender como resolver cada problema ayuda mucho para mejorar la lógica con la que se entiende un problema.

## Bibliografías


[1] R. Kanchan. (2017, Nov. 15) "fgets() and gets() in C language" [Internet]. Disponible en https://www.geeksforgeeks.org/fgets-gets-c-language/

[2] Universidad Nacional de Quilmes (s.f.) "Programación en C con Memoria Dinámica" Disponible en https://sites.google.com/site/programacioniiuno/temario/unidad-1---manejo-de-memoria-dinmica/programacin-en-c-con-memoria-dinmica

[3] (s.f.) "C Dynamic Memory Allocation" Disponible en https://www.programiz.com/c-programming/c-dynamic-memory-allocation

[4] Wikibooks (2019, Sep. 27) "Programación en C/Manejo dinámico de memoria" Disponible en https://es.wikibooks.org/wiki/Programaci%C3%B3n_en_C/Manejo_din%C3%A1mico_de_memoria

[5] Tutorials Point (s.f.) "C library function - sscanf()" Disponible en https://www.tutorialspoint.com/c_standard_library/c_function_sscanf.htm

[6] (2010, Ago. 8) "Multiple initialization in C# 'for' loop" Disponible en https://stackoverflow.com/questions/1658557/multiple-initialization-in-c-sharp-for-loop