

Proyecto Final C

Frida Hernández Rodríguez y Cristopher Velasco Ávila

Mayo 2025

Índice

1. Introducción	3
2. Desarrollo	4
2.1. Definición de Estructuras	4
2.1.1. Imagen del código	4
2.2. Función: <code>crearProducto()</code>	5
2.2.1. Imagen del código	5
2.3. Función: <code>cargarProductos()</code>	6
2.3.1. Imagen del código	6
2.4. Función: <code>mostrarProducto()</code>	6
2.4.1. Imagen del código	7
2.5. Función: <code>agregarAlCarrito()</code>	7
2.5.1. Imagen del código	7
2.6. Función: <code>mostrarCarrito()</code>	7
2.6.1. Imagen del código	8
2.7. Función: <code>navegarProductos()</code>	8
2.7.1. Imagen del código	8
2.8. Función: <code>crearUsuario()</code>	9
2.8.1. Imagen del código	9
2.9. Función: <code>mostrarUsuario()</code>	10
2.9.1. Imagen del código	10
2.10. Función: <code>mostrarMenu()</code>	10
2.10.1. Imagen del código	10
2.11. Función Principal: <code>main()</code>	11
2.11.1. Imagen del código	11
3. Ejecución	12
3.1. Registro de Usuario	13
3.2. Menú Principal	13
3.3. Ver mi carrito de compras - cuando está vacío	13
3.4. Ver lista de productos	14
3.5. Último producto de la lista	14
3.6. Agregar al carrito	14
3.7. Ver mi carrito de compras - con productos	14
3.8. Ver mi información de usuario	15
3.9. Salir de la tienda	16
4. Conclusiones	17
4.1. Conclusión de Frida	17
4.2. Conclusión de Christopher	17

1. Introducción

El presente proyecto fue desarrollado como parte del curso de lenguaje C. El objetivo principal fue implementar un sistema interactivo de compras que simulara el funcionamiento básico de una tiendita virtual. Para ello, se emplearon estructuras, listas ligadas, manejo de archivos y entrada por teclado, así como memoria dinámica mediante el uso de punteros.

El programa permite al usuario registrar su nombre y número celular, navegar una lista de productos cargados desde un archivo .txt, agregar productos a su carrito de compras y visualizar tanto su información personal como el contenido del carrito y el total acumulado a pagar. Todo esto se organiza a través de un menú principal que facilita la interacción con el sistema.

La lógica del sistema se basa en el uso de listas doblemente ligadas para representar tanto el inventario de productos como el carrito del usuario, permitiendo recorrer los elementos hacia adelante y hacia atrás, y realizar operaciones dinámicas de forma eficiente.

2. Desarrollo

2.1. Definición de Estructuras

En este proyecto se definen dos estructuras fundamentales: **Producto** y **Usuario**. Estas estructuras sirven como molde para crear nodos en listas ligadas que almacenan información relevante del sistema (productos y datos del cliente).

Producto representa un nodo de la lista de productos (y también del carrito del usuario). Contiene:

- nombre: el nombre del producto, como cadena de caracteres.
- precio: el precio en formato decimal.
- siguiente: puntero al siguiente nodo de la lista (enlace hacia adelante).
- anterior: puntero al nodo anterior (enlace hacia atrás).

Esta forma de enlazarlos hace que la lista sea doblemente ligada, lo cual permite recorrerla en ambas direcciones (adelante y atrás).

Usuario representa al cliente que está usando el sistema. Contiene:

- nombre: el nombre del usuario.
- numeroCelular: su número telefónico (10 dígitos).
- totalPagar: la suma total de los precios de los productos que ha agregado al carrito.
- Producto* carrito: un puntero que apunta al inicio de la lista ligada.

2.1.1. Imagen del código

```
// Estructuras
typedef struct Producto {
    char nombre[50];
    float precio;
    struct Producto* siguiente; // Apuntador al siguiente producto
    struct Producto* anterior; // Apuntador al producto anterior
} Producto;

typedef struct Usuario {
    char nombre[50];
    char numeroCelular[10];
    float totalPagar;
    Producto* carrito;
} Usuario;
```

Figura 1: Estructura Producto y Usuario

2.2. Función: crearProducto()

En esta sección explicaremos detalladamente la función crearProducto.

Producto*: esta función devuelve un puntero a un Producto.

crearProducto : nombre de la función. **char* nombre:** recibe una cadena (nombre del producto) como parámetro.

float precio: recibe el precio del producto como parámetro.

Producto* nuevo: se declara el puntero nuevo a la estructura/nodo Producto. **malloc(sizeof(Producto)):** se reserva memoria en el heap suficiente para un producto. **(Producto*):** se hace un cast explícito para convertir lo que devuelve **malloc()** a tipo **Producto***. Se verifica si **malloc()** no pudo reservar memoria. Si es así, se imprime un error y se termina el programa con **exit(1)**.

Después, se copia el contenido de la cadena **nombre** dentro del campo **nombre** del nodo **nuevo**. En seguida, se copia el valor del parámetro **precio** al campo **precio** del nodo **nuevo**. Luego, se inicializa el puntero **siguiente** del nodo **nuevo** como **NULL** porque en el momento en que se crea el nodo, todavía no se sabe con qué nodo lo vamos a enlazar. Con el puntero **anterior** haces lo mismo que con el puntero **siguiente**. Esto permite que nos podamos mover hacia adelante y hacia atrás. Y **return nuevo;** devuelve el puntero nuevo al nodo acabamos de crear.

2.2.1. Imagen del código

```
// Gestión de productos

Producto* crearProducto(char* nombre, float precio) {
    Producto* nuevo = (Producto*)malloc(sizeof(Producto));
    if (nuevo == NULL) {
        printf("Error al asignar memoria.\n");
        exit(1);
    }
    strcpy(nuevo->nombre, nombre);
    nuevo->precio = precio;
    nuevo->siguiente = NULL;
    nuevo->anterior = NULL;
    return nuevo;
}
```

Figura 2: Imagen del código

2.3. Función: cargarProductos()

Lee línea por línea un archivo .txt que contiene productos. Por cada línea, crea un nodo con `crearProducto()` y lo agrega al final de una lista doblemente ligada. La variable `inicio` almacena el primer nodo de la lista (la cabeza), y `anterior` se usa para enlazar los nodos sucesivos. Al final, la función retorna el inicio de la lista de productos.

2.3.1. Imagen del código

```
// Leer el archivo txt y listar
Producto* cargarProductos(const char* archivo) {
    FILE* f = fopen(archivo, "r");
    if (f == NULL) {
        printf("Error al abrir el archivo.\n");
        return NULL;
    }

    Producto* inicio = NULL;
    Producto* anterior = NULL;
    char nombre[50];
    float precio;

    while (fscanf(f, "%s %f", nombre, &precio) != EOF) {
        Producto* nuevo = crearProducto(nombre, precio); //Nuevo nodo
        if (inicio == NULL) {
            inicio = nuevo;
        } else {
            anterior->siguiente = nuevo;
            nuevo->anterior = anterior;
        }
        anterior = nuevo;
    }

    fclose(f);
    return inicio;
}
```

Figura 3: Imagen del código

2.4. Función: mostrarProducto()

Recibe un puntero a un nodo de producto y muestra su nombre y precio en consola. Se usa dentro de la navegación de productos para mostrar el nodo actual.

2.4.1. Imagen del código

```
// Mostrar producto
void mostrarProducto(Producto* p) {
    printf("Nombre: %s\n", p->nombre);
    printf("Precio: $%.2f\n", p->precio);
}
```

Figura 4: Imagen del código

2.5. Función: agregarAlCarrito()

Crea un nuevo nodo copiando los datos del `productoActual` (el nodo que el usuario está viendo en la tienda). Inserta ese nuevo nodo al inicio de la lista `usuario->carrito`, y además incrementa el total a pagar. Esta función permite que el carrito tenga su propia lista de productos sin afectar la lista principal.

2.5.1. Imagen del código

```
void agregarAlCarrito(Usuario* usuario, Producto* productoActual) {
    // Nuevo nodo con la información del producto actual
    Producto* nuevo = (Producto*)malloc(sizeof(Producto));
    if (nuevo == NULL) {
        printf("Error al asignar memoria para el carrito.\n");
        exit(1);
    }

    strcpy(nuevo->nombre, productoActual->nombre); // copiar el nombre
    nuevo->precio = productoActual->precio; // copiar precio
    nuevo->siguiente = usuario->carrito; // enlazar el nuevo nodo al inicio del carrito
    nuevo->anterior = NULL;

    if (usuario->carrito != NULL) {
        usuario->carrito->anterior = nuevo;
    }

    usuario->carrito = nuevo; // el carrito empieza con ese producto
    usuario->totalPagar += nuevo->precio; // lo suma al total

    printf("Producto agregado al carrito: %s (%.2f)\n", nuevo->nombre, nuevo->precio);
    getchar(); getchar();
}
```

Figura 5: Imagen del código

2.6. Función: mostrarCarrito()

Muestra el contenido del carrito del usuario recorriendo la lista ligada `usuario.carrito`. Imprime todos los productos con su nombre, precio y un contador, seguido del total acumulado. Si el carrito está vacío, muestra un mensaje.

2.6.1. Imagen del código

```
void mostrarCarrito(Usuario usuario) {
    if (usuario.carrito == NULL) {
        printf("Tu carrito de compras está vacío.\n");
        printf("Puedes volver al menú y agregar productos con la opción 3.\n");
    } else {
        printf("=== Carrito de Compras ===\n");
        Producto* actual = usuario.carrito;
        int contador = 1;
        while (actual != NULL) {
            printf("%d. %s - $%.2f\n", contador, actual->nombre, actual->precio);
            actual = actual->siguiente;
            contador++;
        }
        printf("\nTotal a pagar: $%.2f\n", usuario.totalPagar);
    }
    getchar(); getchar();
}
```

Figura 6: Imagen del código

2.7. Función: navegarProductos()

Permite recorrer la lista de productos uno por uno. Usa teclas:

- **S**: para ir al siguiente nodo
- **A**: para ir al anterior
- **C**: para agregar el producto al carrito
- **Q**: para regresar al menú principal

Se utiliza un ciclo infinito (`while(1)`) hasta que se presione 'Q'. El nodo actual se va actualizando con `actual = actual->siguiente` o `actual->anterior`.

2.7.1. Imagen del código


```
// Menú de navegación
void navegarProductos(Producto* actual, Usuario* usuario) {
    char tecla;
    while (1) {
        system("clear");
        printf("Presiona 'S' para siguiente, 'A' para anterior, 'C' para agregar al carrito, 'Q' para salir.\n");
        mostrarProducto(actual);

        scanf("%c", &tecla);

        if (tecla == 's' || tecla == 'S') {
            if (actual->siguiente != NULL) {
                actual = actual->siguiente;
            } else {
                printf("Ya estás en el último producto.\n"); // Control de desplazamiento si es que llegamos al final de la lista
                getchar(); getchar();
            }
        } else if (tecla == 'a' || tecla == 'A') {
            if (actual->anterior != NULL) {
                actual = actual->anterior;
            } else {
                printf("Ya estás en el primer producto.\n");
                getchar(); getchar();
            }
        } else if (tecla == 'c' || tecla == 'C') {
            agregarAlCarrito(usuario, actual);
        } else if (tecla == 'q' || tecla == 'Q') {
            break;
        }
    }
}
```

Figura 7: Imagen del código

2.8. Función: crearUsuario()

Solicita al usuario su nombre y número celular. Luego inicializa el total a pagar en cero y el carrito en NULL. Devuelve la estructura `Usuario` ya inicializada. Se ejecuta una sola vez al inicio del programa.

2.8.1. Imagen del código

```
// Gestión de usuario
Usuario crearUsuario() {
    Usuario u;

    printf("=== Registro de usuario ===\n");

    printf("Ingresa tu nombre: ");
    scanf("%[^\n]", u.nombre); // lee espacios hasta Enter
    printf("Ingresa tu número celular (10 dígitos): ");
    scanf("%10s", u.numeroCelular); // hasta 10 caracteres

    u.totalPagar = 0.0;
    u.carrito = NULL;

    printf("\n¡Registro exitoso!\n");
    getchar(); getchar(); // pausa para leer

    return u;
}
```

Figura 8: Imagen del código

2.9. Función: mostrarUsuario()

Esta función sirve para mostrar a información del usuario registrado. Recibe una copia de la estructura `Usuario` como parámetro (no es necesario modificarlo, por eso no se pasa por puntero). Imprime tres campos del usuario: `u.nombre`: la cadena con el nombre del usuario. `u.numeroCelular`: el número que ingresó al principio del programa. `u.totalPagar`: el total acumulado por los productos agregados al carrito, mostrado con dos decimales. Esta función se usa cuando el usuario elige la opción 2 del menú principal.

2.9.1. Imagen del código

```
void mostrarUsuario(Usuario u) {
    printf("Nombre: %s\n", u.nombre);
    printf("Número: %s\n", u.numeroCelular);
    printf("Total a pagar: $%.2f\n", u.totalPagar);
}
```

Figura 9: Imagen del código

2.10. Función: mostrarMenu()

Muestra el menú principal del programa. Es una función muy sencilla pero muy útil, porque centraliza la impresión del menú, haciendo que el `main()` sea más limpio y más fácil de leer. Cada `printf` imprime una línea con una opción. Cuando se muestra este menú, el programa espera que el usuario ingrese un número del 1 al 4, que se leerá después en el `main()`.

2.10.1. Imagen del código

```
// Menú principal

void mostrarMenu() {
    printf("=== Menú Principal ===\n");
    printf("1. Ver mi carrito de compras\n");
    printf("2. Ver mi información de usuario\n");
    printf("3. Ver la lista de productos\n");
    printf("4. Salir\n");
}
```

Figura 10: Imagen del código

2.11. Función Principal: `main()`

Es el punto de entrada del programa. Aquí se realiza lo siguiente:

- Se carga la lista de productos desde `productos.txt`.
- Se crea el usuario.
- Se entra en un ciclo que muestra el menú principal.
- Dependiendo de la opción elegida:
 - Se muestra el carrito.
 - Se muestra la información del usuario.
 - Se navega por los productos.
 - Se sale del programa.

Antes de salir, libera la memoria de la lista de productos usando un ciclo con `free()`.

2.11.1. Imagen del código

```

int main() {
    Producto* inicio = cargarProductos("productos.txt");
    Producto* actual = inicio;
    Producto* temp;
    Usuario usuario = crearUsuario();
    int opcion;

    if (actual == NULL) {
        printf("No se cargaron productos.\n");
        return 1;
    }

    while (1) {
        system("clear");
        mostrarMenu();
        scanf("%d", &opcion);

        switch (opcion) {
            case 1:
                mostrarCarrito(usuario);
                break;

            case 2:
                mostrarUsuario(usuario);
                getchar(); getchar();
                break;

            case 3:
                navegarProductos(actual, &usuario);
                break;

            case 4:
                printf("Saliendo de la tiendita...\n");

                // Liberar memoria
                Producto* temp;
                while (inicio != NULL) {
                    temp = inicio;
                    inicio = inicio->siguiente;
                    free(temp);
                }
                return 0;

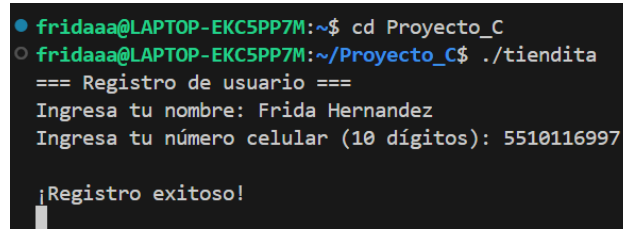
            default:
                printf("Opción no válida.\n");
                getchar(); getchar();
                break;
        }
    }
}

```

Figura 11: Imagen del código

3. Ejecución

3.1. Registro de Usuario

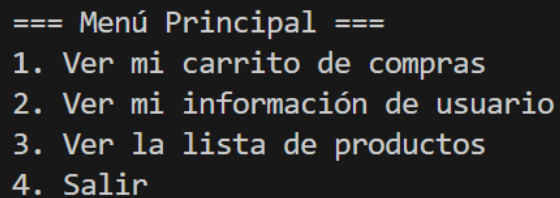


```
fridaaa@LAPTOP-EKC5PP7M:~$ cd Proyecto_C
fridaaa@LAPTOP-EKC5PP7M:~/Proyecto_C$ ./tiendita
=== Registro de usuario ===
Ingresa tu nombre: Frida Hernandez
Ingresa tu número celular (10 dígitos): 5510116997

¡Registro exitoso!
```

Figura 12: Imagen de la ejecución

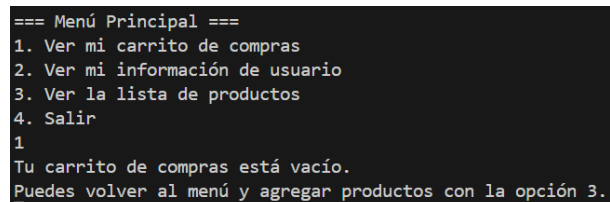
3.2. Menú Principal



```
=== Menú Principal ===
1. Ver mi carrito de compras
2. Ver mi información de usuario
3. Ver la lista de productos
4. Salir
```

Figura 13: Imagen de la ejecución

3.3. Ver mi carrito de compras - cuando está vacío



```
=== Menú Principal ===
1. Ver mi carrito de compras
2. Ver mi información de usuario
3. Ver la lista de productos
4. Salir
1
Tu carrito de compras está vacío.
Puedes volver al menú y agregar productos con la opción 3.
```

Figura 14: Imagen de la ejecución

3.4. Ver lista de productos

```
Presiona 'S' para siguiente, 'A' para anterior, 'C' para agregar al carrito, 'Q' para salir.  
Nombre: CocaCola  
Precio: $16.30  
█
```

Figura 15: Imagen de la ejecución

3.5. Ultimo producto de la lista

```
Presiona 'S' para siguiente, 'A' para anterior, 'C' para agregar al carrito, 'Q' para salir.  
Nombre: Vino  
Precio: $234.00  
s  
Ya estás en el último producto.  
█
```

Figura 16: Imagen de la ejecución

3.6. Agregar al carrito

```
Presiona 'S' para siguiente, 'A' para anterior, 'C' para agregar al carrito, 'Q' para salir.  
Nombre: Café  
Precio: $80.00  
c  
Producto agregado al carrito: Café (80.00)  
█
```

Figura 17: Imagen de la ejecución

3.7. Ver mi carrito de compras - con productos

```
=== Menú Principal ===
1. Ver mi carrito de compras
2. Ver mi información de usuario
3. Ver la lista de productos
4. Salir
1
=== Carrito de Compras ===
1. Vino - $234.00
2. Shampoo - $49.00
3. Queso - $35.90
4. Galletas - $19.00
5. Café - $80.00

Total a pagar: $417.90
█
```

Figura 18: Imagen de la ejecución

3.8. Ver mi información de usuario

```
=== Menú Principal ===
1. Ver mi carrito de compras
2. Ver mi información de usuario
3. Ver la lista de productos
4. Salir
2
Nombre: Frida Hernandez
Número: 551011699
Total a pagar: $417.90
█
```

Figura 19: Imagen de la ejecución

3.9. Salir de la tiendita

```
=== Menú Principal ===  
1. Ver mi carrito de compras  
2. Ver mi información de usuario  
3. Ver la lista de productos  
4. Salir  
4  
Saliendo de la tiendita...
```

Figura 20: Imagen de la ejecución

4. Conclusiones

4.1. Conclusión de Frida

Este proyecto representó un gran reto y, al mismo tiempo, una excelente oportunidad para poner en práctica varios de los conceptos que aprendimos durante el curso, especialmente aquellos que al inicio me parecían abstractos, como las listas ligadas, los nodos, el manejo de memoria dinámica y el uso de estructuras con punteros.

Una de las partes más complejas fue entender cómo se conectan los nodos dentro de una lista doblemente ligada y por qué es necesario crear copias cuando se manejan listas separadas, como en el caso del carrito de compras.

Otro aspecto valioso fue aprender la importancia de liberar correctamente la memoria, una tarea que muchas veces se pasa por alto, pero que es crucial para evitar errores a largo plazo. Comprendí cómo se reserva y se libera memoria manualmente en C.

En general, este proyecto me ayudó a desarrollar habilidades técnicas, de organización del código y de pensamiento lógico. Más allá de cumplir con los requerimientos, me permitió experimentar, equivocarme, corregir y comprender mejor cómo se construyen programas que no solo funcionan, sino que tienen sentido estructuralmente. Sin duda, me deja mejor preparada para seguir aprendiendo C y enfrentar proyectos más complejos en el futuro.

4.2. Conclusión de Cristopher

Este proyecto me ayudó a reforzar varios conceptos importantes de programación en C, especialmente el uso de estructuras, punteros y listas ligadas. Fue interesante ver cómo todo lo que aprendimos en clase puede aplicarse a algo tan concreto como simular una tiendita, donde los datos se almacenan, se recorren y se modifican dinámicamente. Comprender cómo se conectan los nodos, cómo se construye una lista doblemente ligada y cómo se maneja la memoria me permitió avanzar mucho en mi comprensión del lenguaje.

Al mismo tiempo, este proyecto representó un reto importante a nivel de organización personal y trabajo en equipo. Estando cerca del final del semestre, con otras entregas y exámenes encima, fue necesario aprender a priorizar, distribuir tiempos y colaborar de forma clara para cumplir con lo acordado. Aunque hubo momentos de presión, logramos avanzar y terminar el proyecto de manera funcional, lo cual me deja satisfecho tanto por lo aprendido como por el compromiso que implicó sacarlo adelante en conjunto.