Documentación Proyecto Inventario

El programa permite gestionar un inventario de productos que se encuentra alojado en un archivo de texto llamado "inventario.txt".
El programa está escrito en lenguaje C y permite al usuario realizar acciones sobre el inventario mediante funciones, las cuales son:

- Mostrar el inventario guardado en el archivo.
- Agregar productos al inventario.
- Eliminar productos al inventario.
- Modificar las características de productos existentes.
- Reiniciar el Inventario (Eliminar todos los productos).
  El manejo se hace desde un menú de opciones donde el usuario puede elegir la accion que necesite mediante una entrada de números.

Librerias

Las librerías que utilizamos fue:

- studio.h: Librería básica de C, esencial para manejo de archivos.
- string.h: Usada para manipular cadenas de caracteres.
- "inventario.h": Contiene las funciones principales del programa.

Programa principal

Definición de variables

```c
    int opcion;
    int totalProductos = Contar();
```

- **opcion:** Determina la función a utilizar.
- **totalproductos**: Usa la función "Contar" para reconocer cuantos productos existe en el inventario.

Descripción de programa

El programa comienza contabilizando y mostrando el total de productos presente en el archivo de texto.
Se maneja mediante un menú que presenta seis opciones correspondientes a las funciones disponibles.

```c
    printf("\n    MENU INVENTARIO   \n");
    printf("1. Mostrar Inventario\n");
    printf("2. Agregar item\n");
    printf("3. Eliminar item\n");
    printf("4. Modificar item\n");
    printf("5. Reiniciar Inventario\n");
    printf("6. Salir\n");
```

la opción se selecciona mediante el ingreso de un valor numérico correspondiente a las opciones presentadas. se escanea desde el teclado.

```c
     printf("Seleccione una opción: ");
     scanf("%d", &opcion);
     getchar(); // Limpia el búfer.
```

Mediante la sentencia *switch case* se selecciona el caso que el usuario eligió.

```c
    switch (opcion) {
                case 1:
                    if(totalProductos == 0){
                        printf("Inventario Vacío.");
                        break;
                    }else{
                        Mostrar();
                        break;
                    };
                case 2:
                    Agregar();
                    totalProductos = Contar();
                    break;
                case 3:
                    Eliminar();
                    totalProductos = Contar();
                    break;
                case 4:
```

```
                Modificar();
                break;
            case 5:
                ReiniciarInventario();
                totalProductos = Contar();
                break;
            case 6:
                printf("Programa Terminado.\n");
                return 0;
            default:
                printf("Opción Inválida\n");
        }
```

- **Caso 1:** Una sentencia condicional que comprueba si existen productos dentro del inventario para pasar a la función de mostrar el inventario.
- **Caso 2 y Caso 3:** Agrega o Elimina un producto dependiendo de su id asignado, la función *Contar()* actualiza el total del producto.
- **Caso 5:** Elimina todo el inventario y actualiza el total de productos a 0.

Archivo de Cabecera

Dentro del Archivo de cabecera tenemos la definición de las funciones que permiten operar en el programa principal, de igual manera está declarada y definida la estructura principal usada para el programa.

Estructura Principal

La estructura llamada *producto* y permite almacenar las siguientes características de un producto: Un identificador único basado en su lugar en el inventario, el nombre del producto, la cantidad de unidades del producto y su precio.

C

```
// Estructura para los productos
typedef struct producto {
    int id;
    char nombre[50];
    int cantidad;
    float precio;
} Producto;
```

- Para fines practicos y de eficiencia en el codigo, se llama *Producto* a la *struct producto* mediante un *typedef*.

Función Contar

Esta función permite contar los números presentes en el archivo de texto del inventario, retorna un valor entero que simboliza los productos en el inventario.

C

```c
int Contar() {
    FILE *archivo = fopen("inventario.txt", "r"); // Abre el archivo en modo lectura

    if (archivo == NULL) { // Si el archivo no existe o no puede abrirse
        printf("El inventario está vacío o no se pudo abrir el archivo.\n");
        return 0;
    }

    Producto t; // Variable temporal para leer los productos
    int contador = 0;

    // Lee el archivo y cuenta cuántos productos hay.
    while (fscanf(archivo, "%d,%49[^,],%d,%f\n", &t.id, t.nombre, &t.cantidad, &t.precio) == 4) {
        contador++; // Incrementa el contador por cada producto leído
    }

    fclose(archivo); // Cierra el archivo después de contar
    return contador; // Retorna la cantidad total de productos
}
```

- La manera en que la función realiza la acción es mediante un bucle *while* que permite contar todos los documentos existentes y parar cuando estos ya no existan o estén incompletos.

Función Mostrar

Usada principalmente para mostrar en la termina los productos guardados y sus caracteristicas.

C

```c
void Mostrar(){
    // Apertura del archivo
    FILE *archivo = fopen("inventario.txt", "r");
```

```
    // Verifica si el archivo no se pudo abrir y muestra un mensaje
de error
    if (archivo == NULL) {
        printf("Error al abrir el archivo\n");
        return;
    }

    Producto t; // Temporal para mostrar.

    // Formato de archivo.
    printf("\n-------------------------\n");
    printf("  INVENTARIO\n");
    printf("-------------------------\n");

    // Lectura e impresión de productos.
    while(fscanf(archivo, "%d,%49[^,],%d,%f\n", &t.id, t.nombre,
&t.cantidad, &t.precio) == 4){
        printf("ID: %d\n", t.id);
        printf("Nombre: %s\n", t.nombre);
        printf("Cantidad: %d\n", t.cantidad);
        printf("Precio: %.2f\n", t.precio);
        printf("-------------------------\n");
    }

    // Cierre de archivo.
    fclose(archivo);
}
```

- La Función muestra en pantalla el inventario con un formato de encabezado y en lista.
- Mediante el *fscanf* la función escanea las caracteristicas y las asigna a la variable definida llamada *Producto t* donde se almacenan las diferentes características de un producto.

Función Agregar

Como su nombre lo dice, su objetivo es agregar datos correspondientes a un producto al archivo de texto.

C

```
void Agregar() {
    Producto p; // Variable temporal para almacenar los datos del
producto a agregar.
    FILE *archivo = fopen("inventario.txt", "a"); // Abre el archivo
en modo append para agregar al final.
```

```
    if (archivo == NULL) { // Si el archivo no puede abrirse, muestra
un mensaje de error.
        printf("No se pudo abrir el archivo.\n");
        return;
    }

     // Asignar el ID automáticamente basado en la cantidad de
productos existentes
    int cantidadProductos = Contar();
    p.id = cantidadProductos + 1;

    printf("Ingrese el nombre del producto: ");
    fgets(p.nombre, sizeof(p.nombre), stdin); // Lee el nombre
ingresado, incluyendo espacios
    p.nombre[strcspn(p.nombre, "\n")] = '\0'; // Elimina el salto de
línea que agrega `fgets`

    printf("Ingrese la cantidad del producto: ");
    scanf("%d", &p.cantidad); // Lee la cantidad ingresada.

    printf("Ingrese el precio del producto: ");
    scanf("%f", &p.precio); // Lee el precio ingresado

    // Escribe los datos del producto en el archivo
    fprintf(archivo, "%d,%s,%d,%.2f\n", p.id, p.nombre, p.cantidad,
p.precio);
    fclose(archivo); // Cierra el archivo después de escribir
    printf("Producto agregado exitosamente.\n");
}
```

- Se define la variable *p* para almacenar los datos que se escaneen de el texto.
- El archivo es abierto por *fopen()* en modo *append (a)* para poder almacenar el nuevo producto al final del archivo.
- La variable cantidadProductos se actualiza con la función *Contar()* y para la asignación de un identificador de producto nuevo directamente.

C

```
    int cantidadProductos = Contar();
     p.id = cantidadProductos + 1;
```

- Las caracteristicas se añaden con el uso de un *fgets()* para el nombre por su naturaleza de string, para la cantidad y el precio se usa un *scanf()* capaz de asignar a sus respectivos lugares los datos ingresados por el usuario.

```c
    printf("Ingrese el nombre del producto: ");
    fgets(p.nombre, sizeof(p.nombre), stdin); // Lee el nombre ingresado, incluyendo espacios
    p.nombre[strcspn(p.nombre, "\n")] = '\0'; // Elimina el salto de línea que agrega `fgets`

    printf("Ingrese la cantidad del producto: ");
    scanf("%d", &p.cantidad); // Lee la cantidad ingresada.

    printf("Ingrese el precio del producto: ");
    scanf("%f", &p.precio); // Lee el precio ingresado
```

- Pasa a escribir los datos asignados por el usuario en el documento de texto, cierra el archivo e imprime un mensaje de operación exitosa.

Función Modificar

Permite editar las características (Nombre, cantidad, precio) de un producto existente en el archivo.
Mediante la creación de un archivo temporal al que se le escribirán los datos editados por el usuario, que después de acabar la modificación se estaría asignando en lugar del documento original, como un archivo actualizado.

```c
void Modificar() {
    Producto p; // Variable temporal para almacenar los datos de cada producto
    FILE *archivo = fopen("inventario.txt", "r"); // Abre el archivo en modo lectura
    FILE *temp = fopen("temp.txt", "w"); // Crea un archivo temporal en modo escritura
    int idModificar, encontrado = 0; // Variables para el ID del producto a modificar y una bandera
    int opc;

    if (archivo == NULL) { // Si el archivo no puede abrirse, muestra un mensaje de error
        printf("No se pudo abrir el archivo original.\n");
        return;
    }

    if (temp == NULL) { // Si el archivo temporal no puede crearse, muestra un mensaje de error
        printf("No se pudo crear el archivo temporal.\n");
        fclose(archivo);
```

```
            return;
        }

        printf("Ingrese el ID del producto a modificar: "); // Solicita
    el ID del producto a modificar
        scanf("%d", &idModificar); // Lee el ID ingresado
        getchar(); // Limpia el búfer de entrada

        // Recorre el archivo y busca el producto con el ID especificado.
        while (fscanf(archivo, "%d,%49[^,],%d,%f\n", &p.id, p.nombre,
    &p.cantidad, &p.precio) == 4) {
            if (p.id == idModificar) { // Si el ID coincide con el
    producto buscado
                encontrado = 1; // Marca que el producto fue encontrado

                printf("Producto actual: ID=%d, Nombre=%s, Cantidad=%d,
    Precio=%.2f\n",
                       p.id, p.nombre, p.cantidad, p.precio);
                do
                {
                    printf("Selecciona el dato a modificar: \n");
                    printf("1.Nombre\n"
                           "2. Cantidad\n"
                           "3. Precio\n");
                    scanf("%i", &opc);
                    getchar();

                    //Menú interno para modificación.

                    switch(opc){
                        case 1:
                            printf("Ingrese el nuevo nombre del producto:
    ");

                            fgets(p.nombre, sizeof(p.nombre), stdin);
                            p.nombre[strcspn(p.nombre, "\n")] = '\0';
                            break;

                        case 2:
                            printf("Ingrese la nueva cantidad: ");
                            scanf("%d", &p.cantidad);
                            break;

                        case 3:
                             printf("Ingrese el nuevo precio: ");
                            scanf("%f", &p.precio);
                            break;

                        default:
                            printf("Opción no disponible; ingrese
    nuevamente");
```

```
                        break;
                }

            } while (opc < 1 || opc > 3);

        }

        // Escribe los datos (modificados o no) en el archivo
temporal
        fprintf(temp, "%d,%s,%d,%.2f\n", p.id, p.nombre, p.cantidad,
p.precio);
    }

    fclose(archivo); // Cierra el archivo original
    fclose(temp); // Cierra el archivo temporal

    if (encontrado) { // Si se encontró el producto
        remove("inventario.txt"); // Elimina el archivo original
        rename("temp.txt", "inventario.txt"); // Renombra el archivo
temporal al original
        printf("Producto modificado exitosamente.\n");
    } else { // Si no se encontró el producto
        remove("temp.txt"); // Elimina el archivo temporal
        printf("Producto con ID %d no encontrado.\n", idModificar);
    }
}
```

- La función contiene la busqueda del producto dentro del documento de texto y la confirmación de su existencia.

C

```
         printf("Ingrese el ID del producto a modificar: "); //
Solicita el ID del producto a modificar
    scanf("%d", &idModificar); // Lee el ID ingresado
    getchar(); // Limpia el búfer de entrada

    // Recorre el archivo y busca el producto con el ID especificado.
    while (fscanf(archivo, "%d,%49[^,],%d,%f\n", &p.id, p.nombre,
&p.cantidad, &p.precio) == 4) {
        if (p.id == idModificar) { // Si el ID coincide con el
producto buscado
            encontrado = 1; // Marca que el producto fue encontrado

printf("Producto actual: ID=%d, Nombre=%s, Cantidad=%d,
Precio=%.2f\n",p.id, p.nombre, p.cantidad, p.precio);
```

- La función de igual manera comtiene un menú interno en el que se puede elegir que caracteristica especifica del producto se modifica. mediante la variable *opc* esto se lleva a cabo.

C

```c
    do{
            printf("Selecciona el dato a modificar: \n");
            printf("1.Nombre\n"
                           "2. Cantidad\n"
                           "3. Precio\n");
            scanf("%i", &opc);
            getchar();

            //Menú interno para modificación.

            switch(opc){
                    case 1:
                            printf("Ingrese el nuevo nombre del producto:
    ");
                            fgets(p.nombre, sizeof(p.nombre), stdin);
                            p.nombre[strcspn(p.nombre, "\n")] = '\0';
                            break;

                    case 2:
                            printf("Ingrese la nueva cantidad: ");
                            scanf("%d", &p.cantidad);
                            break;

                    case 3:
                             printf("Ingrese el nuevo precio: ");
                            scanf("%f", &p.precio);
                            break;

                    default:
                            printf("Opción no disponible; ingrese
    nuevamente");
                            break;
            }

    } while (opc < 1 || opc > 3);
```

- Funciona mediante un bucle *do-while* que permite que el menpu se repita hasta que el usuario haga un cambio válido, descrito como que *opc* debe estar entre las 3 opciones disponibles representados del 1 al 3.
- En el programa, de haberse modificado con exito, el rpograma escribe dentro del documento temporal con *fprintf()*.

- En la parte final dependiendo del valor de *Encontrado* el programa define si mantener el archivo original o borrarlo y reemplazarlo con el actualizado.

Función Eliminar

Permite eliminar un producto del inventario en el archivo de texto, mediante la creación de un documento temporal donde se actualizaría el documento sin el producto seleccionado.

C

```c
void Eliminar() {
    Producto p;
    FILE *archivo = fopen("inventario.txt", "r");
    FILE *temp = fopen("temp.txt", "w");
    int idEliminar, eliminado = 0, nuevoID = 1;

    if (archivo == NULL) {
        printf("No se pudo abrir el archivo original.\n");
        return;
    }

    if (temp == NULL) {
        printf("No se pudo crear el archivo temporal.\n");
        fclose(archivo);
        return;
    }

    printf("Ingrese el ID del producto a eliminar: "); // Solicita el ID del producto a eliminar
    scanf("%d", &idEliminar); // Lee el ID ingresado
    getchar(); // Limpia el búfer de entrada

    // Recorre el archivo original y copia al temporal los productos que no se van a eliminar
    while (fscanf(archivo, "%d,%49[^,],%d,%f\n", &p.id, p.nombre, &p.cantidad, &p.precio) == 4) {
        if (p.id == idEliminar) { // Si el ID coincide con el producto a eliminar
            eliminado = 1; // Marca que el producto fue eliminado
            printf("Producto eliminado: ID=%d, Nombre=%s, Cantidad=%d, Precio=%.2f\n",
                   p.id, p.nombre, p.cantidad, p.precio);
        } else { // Si el producto no coincide, lo escribe en el archivo temporal
            fprintf(temp, "%d,%s,%d,%.2f\n", nuevoID++, p.nombre, p.cantidad, p.precio);
        }
```

```
    }

    fclose(archivo); // Cierra el archivo original
    fclose(temp); // Cierra el archivo temporal

    if (eliminado) { // Si se eliminó un producto:
        remove("inventario.txt"); // Elimina el archivo original
        rename("temp.txt", "inventario.txt"); // Renombra el archivo
temporal al original
        printf("Producto eliminado y IDs reenumerados
exitosamente.\n");
    } else { // Si no se encontró el producto:
        remove("temp.txt"); // Elimina el archivo temporal
        printf("Producto con ID %d no encontrado.\n", idEliminar);
    }
}
```

- Se utilizan tres variables que describen el id que queremos eliminar(*idEliminar*), una verificación de eliminación(*eliminado*), y un nuevo id encargado de modificar el id de los productos existentes.
- La función solicita al usuario el ID del producto que desea eliminar y luego la función escanea todos los productos en busca de una coincidencia, los archivos que no son coincidentes son escritos en el archivo temporal, de ser encontrado el programa muestra el producto y cambia la variable *eliminado* a 1 en representación de que se eliminó.

C

```
        // Recorre el archivo original y copia al temporal los
productos que no se van a eliminar
    while (fscanf(archivo, "%d,%49[^,],%d,%f\n", &p.id, p.nombre,
&p.cantidad, &p.precio) == 4) {
        if (p.id == idEliminar) { // Si el ID coincide con el
producto a eliminar
            eliminado = 1; // Marca que el producto fue eliminado
            printf("Producto eliminado: ID=%d, Nombre=%s,
Cantidad=%d, Precio=%.2f\n",
                   p.id, p.nombre, p.cantidad, p.precio);
        } else { // Si el producto no coincide, lo escribe en el
archivo temporal
            fprintf(temp, "%d,%s,%d,%.2f\n", nuevoID++, p.nombre,
p.cantidad, p.precio);
        }
    }
```

- La función antes de terminar evalúa si existió una eliminación mediante la variable *eliminado* que de ser verdadera, elimina el archivo original y conserva el actualizado. De no existir una eliminación simplemente conserva el archivo original.

C

```
    if (eliminado) { // Si se eliminó un producto:
        remove("inventario.txt"); // Elimina el archivo original
        rename("temp.txt", "inventario.txt"); // Renombra el archivo
temporal al original
        printf("Producto eliminado y IDs reenumerados
exitosamente.\n");
    } else { // Si no se encontró el producto:
        remove("temp.txt"); // Elimina el archivo temporal
        printf("Producto con ID %d no encontrado.\n", idEliminar);
    }
}
```

Función Reiniciar Inventario

Esta función vacía el archivo de texto donde contiene los datos del inventario.

C

```
void ReiniciarInventario() {
    FILE *archivo = fopen("inventario.txt", "w"); // Abre el archivo
en modo escritura para sobrescribirlo
    if (archivo == NULL) { // Si no se puede abrir el archivo:
        printf("No se pudo reiniciar el inventario.\n");
        return;
    }
    fclose(archivo); // Cierra el archivo vacío
    printf("Inventario reiniciado exitosamente.\n"); // Confirma que
el inventario fue reiniciado
}
```

- Usa una característica implícita en la manera en como se maneja el *fopen()*, al momento de ser abierto en modo de escritura, el programa sobrescribe el archivo existente con uno vacío.

Referencias