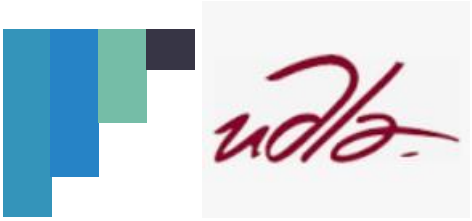


A_ABET_PROYECTO_INTEGRADOR_PMEA_RI2

INFORME

ISWZ1102 – PROGRAMACIÓN 1

Progreso 3

Cristian López

Índice

Introducción……..3

Objetivos del trabajo……3

Tablas de identificación del problema. ……..4

Planteamiento de propuestas de solución ……5

Análisis de las propuestas de solución……5

Selección de la mejor solución……6

Diseño y desarrollo de la solución. ……..6-12

Conclusiones y recomendaciones……12

• Introducción

Como introducción, el presente trabajo, se tomó en cuenta busca solucionar el problema que tiene la Distribuidora Ferretera “JL”, para ello antes hay que dar contexto de ¿qué es esta Distribuidora?,¿Qué hace? ¿Y cómo ha surgido su problema?, en principio es una compañía que se dedica a importar y vender producto del ámbito ferretero de manera nacional e internacional, contempla desde construcciones hasta productos industriales. Su problema yace en que no tiene un sistema de inventario digital, es decir, que normalmente trabajan eso de manera arcaica o escrito:

- Uno de los problemas identificados es la ausencia de un sistema de inventario, ya dicho antes, cosa que es muy grave debido a que no se puede registrar los ingresos de productos. Esta carencia viene acompañada de que se necesita en este sistema las funcionalidades como editar el producto ingresado, eliminarlo y listarlo.

- Aparte de ese problema también se identificó otro, y es que un sistema de inventario en una distribuidora es fundamental que tenga más funcionalidades como es ingresar el proveedor, la marca del producto el origen de llegada de este mismo.

• Objetivos del trabajo.

Básicamente, los objetivos del trabajo son claros y precisos, dar solución a los problemas que tiene la Distribuidora, para crear un sistema de inventario, los objetivos específicos del trabajo son:

1. Crear un sistema de inventario que tenga las siguientes funcionalidades:
    - En el menú principal: Ingresar Producto: ingresar el nombre del producto, ingresar la marca del producto, ingresar su cantidad, el precio, el proveedor y el origen, es decir, la ciudad y país de donde proviene.
    - Permitir editar el producto ingresado.
    - Permitir Eliminar el producto.
    - Listar el/los productos ingresados.
    - Salir.

• Tablas de identificación del problema.

Nombre del sistema	
Usuarios	Supervisor de la mercadería.
Objetivos	Crear un sistema de inventario que tenga las siguientes funcionalidades: - En el menú principal: Ingresar Producto: ingresar el nombre del producto, ingresar la marca del producto, ingresar su cantidad, el precio, el proveedor y el origen, es decir, la ciudad y país de donde proviene. - Permitir editar el producto ingresado. - Permitir Eliminar el producto. - Listar el/los productos ingresados. - Salir.
Contexto del problema (Considerar los contextos económico, social y ambiental, dentro del sector productivo en el que funcionaría el sistema de inventarios)	Contexto económico: • El sector ferretero es parte fundamental de la industria de la construcción y el mantenimiento, lo que implica una alta demanda de productos ferreteros. • El sistema de inventario digital permitirá a la distribuidora mejorar la gestión de su inventario, optimizando los niveles de stock y evitando pérdidas por productos obsoletos o vencidos. Contexto social: • La Distribuidora Ferretera "JL" desempeña un papel importante en la cadena de suministro de productos ferreteros, atendiendo a clientes minoristas y mayoristas. Contexto ambiental: • Al optimizar el inventario, se reduce la necesidad de almacenar grandes cantidades de productos, lo que puede ayudar a minimizar el uso de recursos naturales y los residuos

	generados.
Restricciones (Consideraciones externas que se deben tener para diseñar y desarrollar el proyecto, ejemplo: porcentaje de impuesto, registro sanitario u otros)	Registro de Proveedores, lugar de origen del producto, marca.
Limitaciones (Consideraciones internas que se deben tener para diseñar y desarrollar el proyecto, ejemplo: lenguaje de programación, tipos de almacenamiento u otros)	Lenguaje de Programacion: C. Visual Studio Code.

• Planteamiento de propuestas de solución.

Para dar solución a los problemas propuestos y analizados, principalmente se va implementar sistemas, programas o softwares capaces de dar solución a dichos problemas, por lo que se propone dos soluciones:

Soluciones en Github:
https://github.com/ElbergonGod/A_ABET_PROYECTO_INTEGRADOR_PMEA_RI2

• Análisis de propuestas de solución.

- Primera solución:

Este programa es un sistema de inventario que permite ingresar, editar, eliminar y listar productos, es decir, implementa un sistema básico de gestión de inventario que permite al usuario interactuar con los productos almacenados, agregar nuevos productos, editar productos existentes, eliminar productos y ver la lista de productos. Mediante funciones como “void ingresarProductos(), listarproductos, editar productos.etc”. Llamadas después en main, también se usan bucles y punteros.

- Segunda Solución:

Al igual que el anterior, este programa utiliza estructuras y arreglos para almacenar los productos en memoria. Las funciones y estructuras son muy similares, pero con algunos cambios en la forma en que se ingresan y editan los datos.

En esta versión, se utilizan estructuras anidadas, donde Inventario contiene un arreglo de estructuras Producto. Las funciones ingresarProducto, editarProducto y eliminarProducto solicitan al usuario que ingrese los datos correspondientes a cada atributo del producto, como nombre, marca, cantidad, precio, proveedor, ciudad y país de llegada. Estos datos se asignan a una nueva estructura Producto, que luego se agrega al arreglo de productos del inventario.

La función listarProductos muestra los detalles de cada producto ingresado hasta el momento, incluyendo el índice, nombre, marca, cantidad, precio,

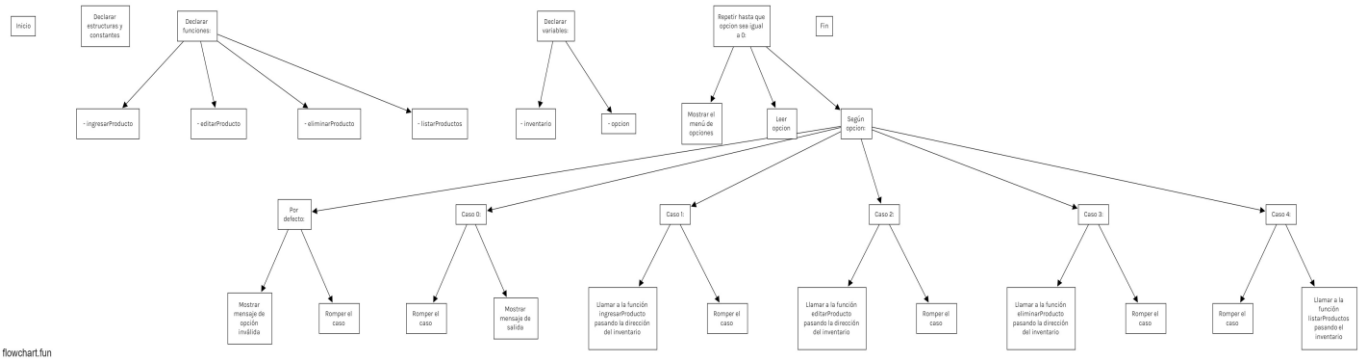
proveedor, ciudad y país de llegada. La principal diferencia entre este programa y el anterior radica en la forma en que se obtienen y almacenan los datos de los productos, pero la lógica general y la funcionalidad siguen siendo las mismas.

• Selección de la mejor solución.

Para la selección, definitivamente se escogió la primera opción, simplemente es mas cómoda al programador al entender el código, mas intuitiva al usuario. Y tienen un mejor orden por archivos.

• Diseño y desarrollo de la solución.

En esta parte, implica como se armará la solución del problema, es decir, el sistema y como cree una solución, por ello diseñe un par de pasos para explicar como el software que se implementará que dará solución a la problemática:
Comenzando por el:
1.- diagrama de flujo (está en github también):



2.- Identificación de variables de entrada y salida.

De entrada: Opción, nombre del producto, marca del producto, cantidad del producto, precio del producto, ciudad y país de donde viene el producto, proveedor.

De salida: Inventario, producto, índice, archivo y nuevo producto.

3.- Enlace de Github al código fuente del programa:

Link: https://github.com/ElbergonGod/A_ABET_PROYECTO_INTEGRADOR_PMEA_RI2

4.- Imágenes de las partes más importantes del código explicadas.

Funciones.c:

```c
void ingresarProducto(struct Inventario *inventario) {
    if (inventario->cantidadProductos < MAX_PRODUCTOS) {
        struct Producto nuevoProducto;
        printf("Ingrese el nombre del producto: ");
        scanf("%s", nuevoProducto.nombre);
        printf("Ingrese la marca del producto: ");
        scanf("%s", nuevoProducto.marca);
        printf("Ingrese la cantidad disponible: ");
        scanf("%d", &nuevoProducto.cantidad);
        printf("Ingrese el precio del producto: ");
        scanf("%f", &nuevoProducto.precio);
        printf("Ingrese el proveedor del producto: ");
        scanf("%s", nuevoProducto.proveedor);
        printf("Ingrese la ciudad de llegada del producto: ");
        scanf("%s", nuevoProducto.ciudad);
        printf("Ingrese el país de llegada del producto: ");
        scanf("%s", nuevoProducto.pais);

        inventario->productos[inventario->cantidadProductos++] = nuevoProducto;
        printf("Producto ingresado exitosamente.\n");

        // Guardar producto en archivo
        FILE *archivo = fopen("productos.txt", "a");
        if (archivo != NULL) {
            fprintf(archivo, "Índice: %d\n", inventario->cantidadProductos - 1);
            fprintf(archivo, "Nombre: %s\n", nuevoProducto.nombre);
            fprintf(archivo, "Marca: %s\n", nuevoProducto.marca);
            fprintf(archivo, "Cantidad: %d\n", nuevoProducto.cantidad);
            fprintf(archivo, "Precio: %.2f\n", nuevoProducto.precio);
            fprintf(archivo, "Proveedor: %s\n", nuevoProducto.proveedor);
            fprintf(archivo, "Ciudad de llegada: %s\n", nuevoProducto.ciudad);
            fprintf(archivo, "País de llegada: %s\n\n", nuevoProducto.pais);
            fclose(archivo);
        } else {
            printf("No se pudo abrir el archivo para guardar el producto.\n");
        }
    } else {
```

- La Función Ingresar Producto: Básicamente, se encarga de agregar un nuevo producto al inventario. Comprueba si la cantidad de productos en el inventario es menor que el límite máximo de productos (MAX_PRODUCTOS). Declara una variable local nuevoProducto del tipo struct Producto. Esta variable almacenará la información del nuevo producto a ingresar. Agrega el nuevo producto al arreglo de productos del inventario inventario->productos[inventario->cantidadProductos++] = nuevoProducto. Incrementa la cantidad de productos en el inventario después de agregar el nuevo producto. Abre el archivo "productos.txt" en modo de escritura "a" (agregar al final del archivo). Si el archivo se abre correctamente, se escriben los detalles del nuevo producto en el archivo utilizando la función “fprintf” y Cierra el archivo.

```c
void editarProducto(struct Inventario *inventario) {
    int indice;
    printf("Ingrese el índice del producto a editar: ");
    scanf("%d", &indice);

    if (indice >= 0 && indice < inventario->cantidadProductos) {
        struct Producto *producto = &inventario->productos[indice];
        printf("Ingrese el nuevo nombre del producto: ");
        scanf("%s", producto->nombre);
        printf("Ingrese la nueva marca del producto: ");
        scanf("%s", producto->marca);
        printf("Ingrese la nueva cantidad disponible: ");
        scanf("%d", &producto->cantidad);
        printf("Ingrese el nuevo precio del producto: ");
        scanf("%f", &producto->precio);
        printf("Ingrese el nuevo proveedor del producto: ");
        scanf("%s", producto->proveedor);
        printf("Ingrese la nueva ciudad de llegada del producto: ");
        scanf("%s", producto->ciudad);
        printf("Ingrese el nuevo país de llegada del producto: ");
        scanf("%s", producto->pais);

        printf("Producto editado exitosamente.\n");
    } else {
        printf("Índice inválido. No se encontró el producto.\n");
    }
}
```

- La función editarProducto permite modificar los detalles de un producto existente en el inventario. Aquí se explica su funcionamiento: Se solicita al usuario que ingrese el índice del producto que desea editar. Se verifica si el índice ingresado es válido, es decir, si se encuentra dentro del rango de productos existentes en el inventario. Si el índice es válido, se procede a realizar la edición del producto. Se obtiene un puntero al producto en la posición indicada por el índice: struct Producto *producto = &inventario->productos[indice];. Esto permite acceder y modificar los datos del producto directamente en el arreglo del inventario.

```c
void eliminarProducto(struct Inventario *inventario) {
    int indice;
    printf("Ingrese el índice del producto a eliminar: ");
    scanf("%d", &indice);

    if (indice >= 0 && indice < inventario->cantidadProductos) {
        for (int i = indice; i < inventario->cantidadProductos - 1; i++) {
            inventario->productos[i] = inventario->productos[i + 1];
        }
        inventario->cantidadProductos--;

        printf("Producto eliminado exitosamente.\n");
    } else {
        printf("Índice inválido. No se encontró el producto.\n");
    }
}
```

- La función eliminarProducto permite eliminar un producto del inventario. A continuación se explica su funcionamiento: Se solicita al usuario que ingrese el índice del producto que desea eliminar. Se verifica si el índice ingresado es válido, es decir, si se encuentra dentro del rango de productos existentes en el inventario. Si el índice es válido, se procede a eliminar el producto. Se realiza un bucle for desde el índice indicado hasta el penúltimo producto del inventario. En cada iteración, se reemplaza el producto en la posición actual con el producto de la siguiente posición, lo cual "desplaza" todos los productos una posición hacia atrás en el arreglo. Se decrementa la cantidad de productos en el inventario: inventario->cantidadProductos--. Se muestra un mensaje indicando que el producto se ha eliminado exitosamente. Si el índice ingresado es inválido, es decir, no se encontró el producto en el inventario, se muestra un mensaje de error.

```c
void listarProductos(struct Inventario inventario) {
    printf("Productos ingresados:\n");
    for (int i = 0; i < inventario.cantidadProductos; i++) {
        struct Producto producto = inventario.productos[i];
        printf("Índice: %d\n", i);
        printf("Nombre: %s\n", producto.nombre);
        printf("Marca: %s\n", producto.marca);
        printf("Cantidad: %d\n", producto.cantidad);
        printf("Precio: %.2f\n", producto.precio);
        printf("Proveedor: %s\n", producto.proveedor);
        printf("Ciudad de llegada: %s\n", producto.ciudad);
        printf("País de llegada: %s\n", producto.pais);
        printf("\n");
    }
}
```

- La función listarProductos muestra por pantalla la lista de productos ingresados en el inventario. Itera a través de cada producto en el arreglo productos del inventario y muestra su información, como el índice, nombre, marca, cantidad, precio, proveedor, ciudad de llegada y país de llegada.
- El propósito de esta función es proporcionar una vista general de todos los productos registrados en el inventario, brindando detalles importantes de cada uno de ellos. La información se muestra en un formato legible y estructurado para facilitar la lectura y comprensión de los datos.
- En resumen, la función listarProductos se encarga de imprimir en la consola los detalles de todos los productos almacenados en el inventario, proporcionando una visualización completa de la información.

Funciones.h:

funciones.h

```c
#ifndef FUNCIONES_H
#define FUNCIONES_H

#define MAX_PRODUCTOS 100

struct Producto {
    char nombre[50];
    char marca[50];
    int cantidad;
    float precio;
    char proveedor[50];
    char ciudad[50];
    char pais[50];
};

struct Inventario {
    struct Producto productos[MAX_PRODUCTOS];
    int cantidadProductos;
};

void ingresarProducto(struct Inventario *inventario);

void editarProducto(struct Inventario *inventario);

void eliminarProducto(struct Inventario *inventario);

void listarProductos(struct Inventario inventario);

#endif /* FUNCIONES_H */
```

- Simplemente se encarga de usar struct, para almacenar las variables declaradas, en Producto, también inventario, y llama las funciones escritas en funciones.c.

Main.c:

main.c

main.c > main()

```c
#include <stdio.h>
#include "funciones.h"

int main() {
    struct Inventario inventario;
    inventario.cantidadProductos = 0;

    int opcion;
    do {
        printf("----- Sistema de Inventario -----\n");
        printf("1. Ingresar producto\n");
        printf("2. Editar producto\n");
        printf("3. Eliminar producto\n");
        printf("4. Listar productos\n");
        printf("0. Salir\n");
        printf("Ingrese una opción: ");
        scanf("%d", &opcion);

        switch (opcion) {
            case 1:
                ingresarProducto(&inventario);
                break;
            case 2:
                editarProducto(&inventario);
                break;
            case 3:
                eliminarProducto(&inventario);
                break;
            case 4:
                listarProductos(inventario);
                break;
            case 0:
                printf("Saliendo del sistema...\n");
                break;
            default:
                printf("Opción inválida. Intente nuevamente.\n");
                break;
```

```
33                  printf("Saliendo del sistema...\n");
34                  break;
35              default:
36                  printf("Opción inválida. Intente nuevamente.\n");
37                  break;
38          }
39
40          printf("\n");
41      } while (opcion != 0);
42
43      return 0;
44  }
45
```

- Es el punto de entrada del programa. En esta función, se realiza la interacción con el usuario a través de un menú para gestionar un sistema de inventario.
  La función main realiza lo siguiente:
  Declara e inicializa una variable inventario de tipo struct Inventario. Esta estructura contiene un arreglo de productos y la cantidad de productos en el inventario.
  Declara una variable opcion para almacenar la opción seleccionada por el usuario en el menú.
  Se inicia un bucle do-while que se ejecutará hasta que el usuario seleccione la opción "0" para salir del programa. Se solicita al usuario que ingrese una opción seleccionando un número del menú.
  Se utiliza una estructura de control switch para ejecutar el código correspondiente a la opción seleccionada por el usuario. Por ejemplo, si el usuario elige la opción "1", se llama a la función ingresarProducto pasando como argumento la dirección del inventario.

  Después de ejecutar la opción seleccionada, se muestra un salto de línea para separar visualmente las iteraciones del menú.

  El bucle se repite hasta que el usuario seleccione la opción "0" para salir del programa.

  Al final, se retorna el valor 0 para indicar que el programa finalizó correctamente.

• Conclusiones y recomendaciones.

Finalmente, se logró solucionar la problemática que tenía la Distribuidora dándole un software que permita satisfacer el problema con el ingreso de productos, por un sistema de inventario.

Como recomendación, se puede mejorar la calidad del mismo con más tiempo y esfuerzo.