UNIVERSIDAD AUTÓNOMA DE BAJA CALIFORNIA

Desarrollo de Software y Tecnologías Emergentes

Lenguaje C

ALUMNO: Barrios Retana Lizeth
MATRÍCULA: 372813
GRUPO: 932

PROFESOR: Yulith Vanessa Altamirano Flores

Ensenada, Baja California a 25 de Noviembre de 2023.

Práctica 8. Estructuras

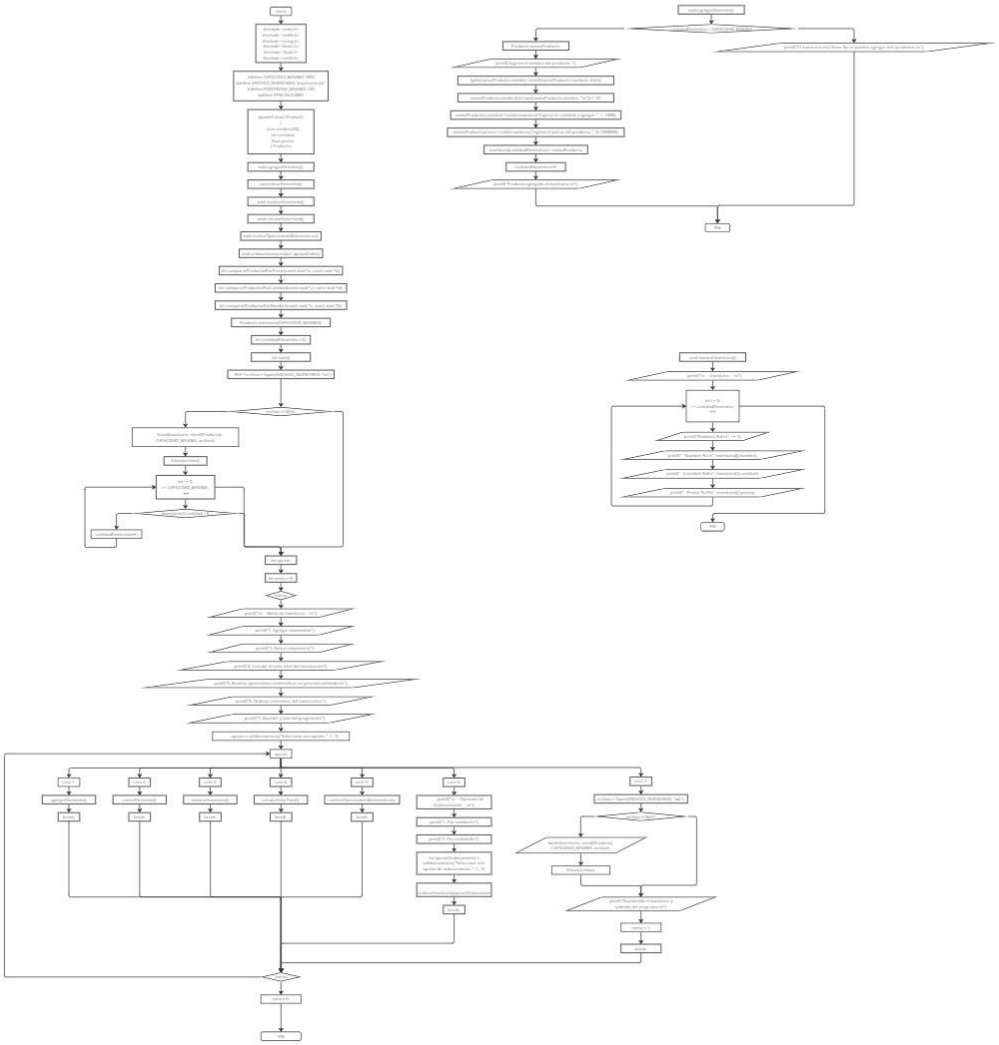
instrucciones

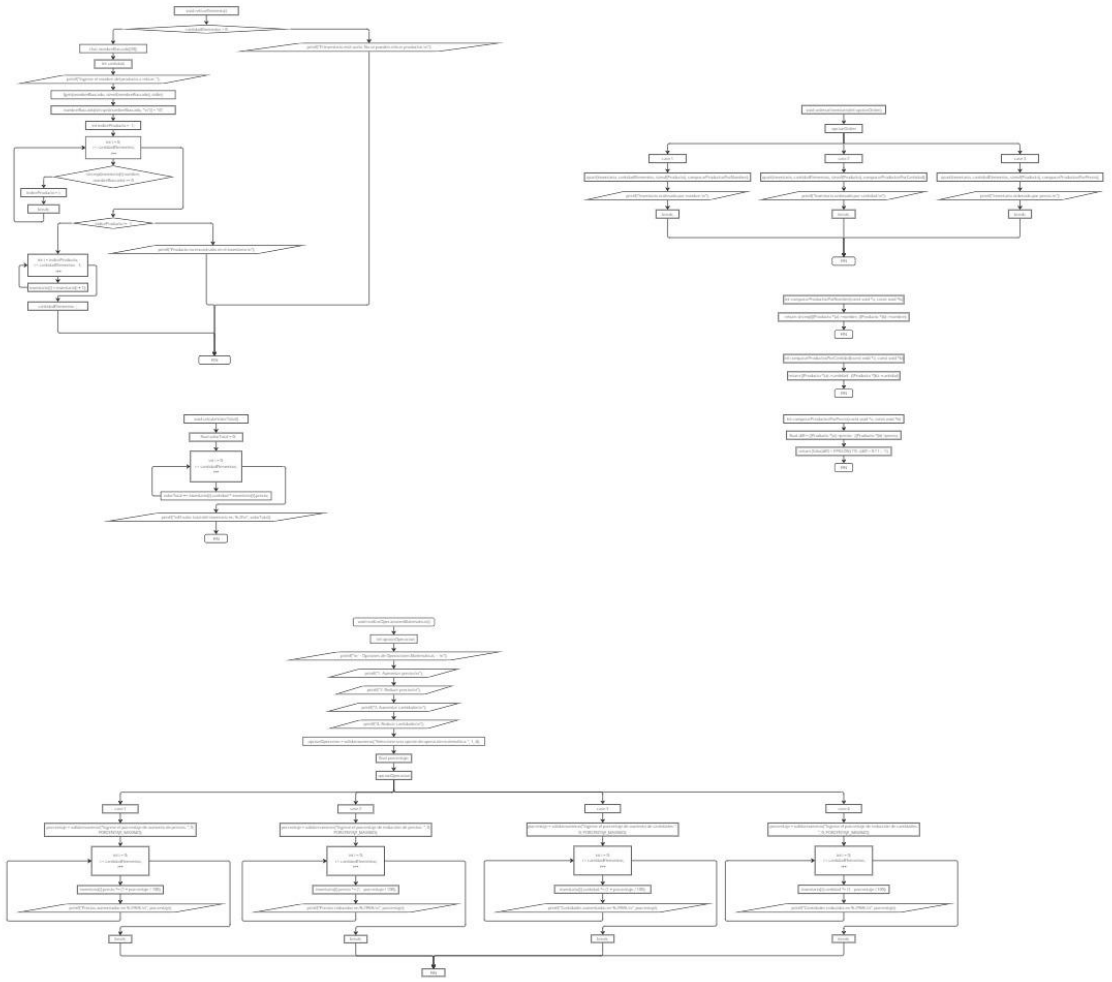
Desarrollen el código en lenguaje C y elaboren el diagrama de flujo correspondiente para los ejercicios. Será suficiente con un archivo .cpp que contenga todos los ejercicios organizados en un menú.

Repositorio

`https://github.com/LizBarrios/Practica8_Estructuras_BarriosRetana_Lizeth.git`

Diagrama de Flujo





Problemas

1. Crear una estructura: Inicializa un inventario vacío con una capacidad máxima utilizando una estructura llamada “Producto” para representar los elementos del inventario. La estructura “Producto” debe incluir campos como nombre, cantidad y precio.
2. Presenta al usuario un menú que le permita realizar las siguientes operaciones:
    - Agregar elementos al inventario: Permite al usuario ingresar el nombre, cantidad y precio del producto y agrega un nuevo elemento al inventario. Asegúrate de manejar situaciones en las que el inventario esté lleno.
    - Retirar elementos del inventario: Permite al usuario ingresar el nombre del producto que desea retirar y elimina ese elemento del inventario. Asegúrate de manejar casos en los que el elemento no esté en el inventario.

- Mostrar el inventario: Muestra al usuario el contenido actual del inventario, incluyendo el nombre, cantidad y precio de cada producto.
- Calcular el valor total del inventario: Agrega una opción al menú que calcule y muestre el valor total del inventario, que es la suma del precio de cada producto multiplicado por su cantidad en stock.
- Salir del programa: Permite al usuario salir del programa cuando lo desee.

3. Operaciones: Implementa un bucle que permita al usuario realizar múltiples operaciones.
4. Productos: Utiliza incrementadores y decrementadores para ajustar la cantidad de productos en el inventario al agregar o retirar elementos.
5. Validaciones: Maneja situaciones en las que el inventario esté lleno o vacío. Asegúrate de validar las operaciones para evitar errores.
6. Agrega una instrucción relacionada con la Práctica 2:
   - Ordenar elementos del inventario: Permite al usuario ordenar los productos en el inventario por nombre, cantidad o precio, según su elección.

7. Considera la posibilidad de agregar características adicionales, como guardar el inventario en un archivo para persistencia de datos entre sesiones y realizar operaciones matemáticas en los precios o la cantidad de productos.

```c
/*
 Nombre del archivo:2.c
 Autor: Lizeth Barrios Retana
 Fecha de creación: 20 de Noviembre de 2023
 Descripción: Este programa simula un sistema de gestion de inventario utilizando
structs donde los usuarios pueden agregar y retirar elementos del inventario
*/


#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <limits.h>
#include <float.h>
#include <math.h>
#include "BRL.h"

#define CAPACIDAD_MAXIMA 1000
#define ARCHIVO_INVENTARIO "inventario.dat"
#define PORCENTAJE_MAXIMO 100
```

```c
#define EPSILON 0.0001

typedef struct Producto
{
    char nombre[30];
    int cantidad;
    float precio;
} Producto;

void agregarElemento();
void retirarElemento();
void mostrarInventario();
void calcularValorTotal();
void realizarOperacionesMatematicas();
void ordenarInventario(int opcionOrden);
int compararProductosPorPrecio(const void *a, const void *b);
int compararProductosPorCantidad(const void *a, const void *b);
int compararProductosPorNombre(const void *a, const void *b);

Producto inventario[CAPACIDAD_MAXIMA];

int cantidadElementos = 0;

int main()
{
    FILE *archivo = fopen(ARCHIVO_INVENTARIO, "rb");
    if (archivo != NULL)
    {
        fread(inventario, sizeof(Producto), CAPACIDAD_MAXIMA, archivo);
        fclose(archivo);

        for (int i = 0; i < CAPACIDAD_MAXIMA; i++)
        {
            if (inventario[i].cantidad > 0)
            {
                cantidadElementos++;
            }
        }
    }

    int opcion;
    int menu = 0;

    do
    {
        printf("\n--- Menú de Inventario ---\n");
        printf("1. Agregar elemento\n");
        printf("2. Retirar elemento\n");
        printf("3. Mostrar inventario\n");
        printf("4. Calcular el valor total del inventario\n");
```

```
        printf("5. Realizar operaciones matemáticas en precios/cantidades\n");
        printf("6. Ordenar elementos del inventario\n");
        printf("7. Guardar y salir del programa\n");

        opcion = validarnumeros("Seleccione una opción: ", 1, 7);
        switch (opcion)
        {
        case 1:
            agregarElemento();
            break;
        case 2:
            retirarElemento();
            break;
        case 3:
            mostrarInventario();
            break;
        case 4:
            calcularValorTotal();
            break;
        case 5:
            realizarOperacionesMatematicas();
            break;
        case 6:
            printf("\n--- Opciones de Ordenamiento ---\n");
            printf("1. Por nombre\n");
            printf("2. Por cantidad\n");
            printf("3. Por precio\n");
               int opcionOrdenamiento = validarnumeros("Seleccione una opción de
ordenamiento: ", 1, 3);
            ordenarInventario(opcionOrdenamiento);
            break;
        case 7:
            archivo = fopen(ARCHIVO_INVENTARIO, "wb");
            if (archivo != NULL)
            {
                fwrite(inventario, sizeof(Producto), CAPACIDAD_MAXIMA, archivo);
                fclose(archivo);
            }

            printf("Guardando el inventario y saliendo del programa.\n");
            menu = 1;
            break;
        }
    } while (!menu);

    return 0;
}

/*
 Función: agregarElemento
```

```
 Descripción: Agrega un nuevo producto al inventario.
 Parámetros: Ninguno
 Valor de retorno: Ninguno
*/
void agregarElemento()
{
    if (cantidadElementos < CAPACIDAD_MAXIMA)
    {
        Producto nuevoProducto;

        printf("Ingrese el nombre del producto: ");
        fgets(nuevoProducto.nombre, sizeof(nuevoProducto.nombre), stdin);
        nuevoProducto.nombre[strcspn(nuevoProducto.nombre, "\n")] = '\0';

          nuevoProducto.cantidad = validarnumeros("Ingrese la cantidad a agregar:
", 1, 1000);
         nuevoProducto.precio = validarnumeros("Ingrese el precio del producto: ",
0, 1000000);

        inventario[cantidadElementos] = nuevoProducto;
        cantidadElementos++;
        printf("Producto agregado al inventario.\n");
    }
    else
    {
                printf("El  inventario  está  lleno.  No  se  pueden  agregar  más
productos.\n");
    }
}

/*
 Función: retirarElemento
 Descripción: Retira un producto del inventario.
 Parámetros: Ninguno
 Valor de retorno: Ninguno
*/
void retirarElemento()
{
    if (cantidadElementos > 0)
    {
        char nombreBuscado[30];
        int cantidad;

        printf("Ingrese el nombre del producto a retirar: ");
        fgets(nombreBuscado, sizeof(nombreBuscado), stdin);
        nombreBuscado[strcspn(nombreBuscado, "\n")] = '\0';

        int indiceProducto = -1;
        for (int i = 0; i < cantidadElementos; i++)
        {
```

```
            if (strcmp(inventario[i].nombre, nombreBuscado) == 0)
            {
                indiceProducto = i;
                break;
            }
        }

        if (indiceProducto != -1)
        {
                cantidad = validarnumeros("Ingrese la cantidad a retirar: ", 1,
inventario[indiceProducto].cantidad);

            inventario[indiceProducto].cantidad -= cantidad;
            printf("Producto retirado del inventario.\n");

            if (inventario[indiceProducto].cantidad == 0)
            {
                for (int i = indiceProducto; i < cantidadElementos - 1; i++)
                {
                    inventario[i] = inventario[i + 1];
                }
                cantidadElementos--;
            }
        }
        else
        {
            printf("Producto no encontrado en el inventario.\n");
        }
    }
    else
    {
        printf("El inventario está vacío. No se pueden retirar productos.\n");
    }
}

/*
 Función: mostrarInventario
 Descripción: Muestra el contenido actual del inventario.
 Parámetros: Ninguno
 Valor de retorno: Ninguno
*/
void mostrarInventario()
{
    printf("\n--- Inventario ---\n");

    for (int i = 0; i < cantidadElementos; i++)
    {
        printf("Producto %d:\n", i + 1);
        printf("   Nombre: %s\n", inventario[i].nombre);
        printf("   Cantidad: %d\n", inventario[i].cantidad);
```

```
        printf("   Precio: %.2f\n", inventario[i].precio);
    }
}

/*
 Función: calcularValorTotal
 Descripción: Calcula el valor total del inventario.
 Parámetros: Ninguno
 Valor de retorno: Ninguno
*/
void calcularValorTotal()
{
    float valorTotal = 0;

    for (int i = 0; i < cantidadElementos; i++)
    {
        valorTotal += inventario[i].cantidad * inventario[i].precio;
    }

    printf("\nEl valor total del inventario es: %.2f\n", valorTotal);
}

/*
 Función: ordenarInventario
 Descripción: Ordena el inventario según la opción seleccionada.
 Parámetros:
 - opcionOrden: Opción de ordenamiento (1: por nombre, 2: por cantidad, 3: por
precio).
 Valor de retorno: Ninguno
*/
void ordenarInventario(int opcionOrden)
{
    switch (opcionOrden)
    {
    case 1:
                        qsort(inventario,   cantidadElementos,   sizeof(Producto),
compararProductosPorNombre);
        printf("Inventario ordenado por nombre.\n");
        break;
    case 2:
                        qsort(inventario,   cantidadElementos,   sizeof(Producto),
compararProductosPorCantidad);
        printf("Inventario ordenado por cantidad.\n");
        break;
    case 3:
                        qsort(inventario,   cantidadElementos,   sizeof(Producto),
compararProductosPorPrecio);
        printf("Inventario ordenado por precio.\n");
        break;
    }
```

```c
}

/*
 Función: compararProductosPorNombre
 Descripción: Función de comparación para ordenar productos por nombre.
 Parámetros:
 - a: Primer elemento a comparar.
 - b: Segundo elemento a comparar.
 Valor de retorno: Entero negativo, cero o positivo según si a es menor, igual o
mayor que b.
*/
int compararProductosPorNombre(const void *a, const void *b)
{
    return strcmp(((Producto *)a)->nombre, ((Producto *)b)->nombre);
}

/*
 Función: compararProductosPorCantidad
 Descripción: Función de comparación para ordenar productos por cantidad.
 Parámetros:
 - a: Primer elemento a comparar.
 - b: Segundo elemento a comparar.
 Valor de retorno: Entero negativo, cero o positivo según si a es menor, igual o
mayor que b.
*/
int compararProductosPorCantidad(const void *a, const void *b)
{
    return ((Producto *)a)->cantidad - ((Producto *)b)->cantidad;
}

/*
 Función: compararProductosPorPrecio
 Descripción: Función de comparación para ordenar productos por precio.
 Parámetros:
 - a: Primer elemento a comparar.
 - b: Segundo elemento a comparar.
 Valor de retorno: Entero negativo, cero o positivo según si a es menor, igual o
mayor que b.
*/
int compararProductosPorPrecio(const void *a, const void *b)
{
    float diff = ((Producto *)a)->precio - ((Producto *)b)->precio;
    return (fabs(diff) < EPSILON) ? 0 : (diff > 0 ? 1 : -1);
}

/*
 Función: realizarOperacionesMatematicas
 Descripción: Permite al usuario realizar operaciones matemáticas en precios y
cantidades.
 Parámetros: Ninguno
```

```
 Valor de retorno: Ninguno
*/
void realizarOperacionesMatematicas()
{
    int opcionOperacion;
    printf("\n--- Opciones de Operaciones Matemáticas ---\n");
    printf("1. Aumentar precios\n");
    printf("2. Reducir precios\n");
    printf("3. Aumentar cantidades\n");
    printf("4. Reducir cantidades\n");
        opcionOperacion = validarnumeros("Seleccione una opción de operación matemática: ", 1, 4);

    float porcentaje;
    switch (opcionOperacion)
    {
    case 1:
         porcentaje = validarnumeros("Ingrese el porcentaje de aumento de precios: ", 0, PORCENTAJE_MAXIMO);
        for (int i = 0; i < cantidadElementos; i++)
        {
            inventario[i].precio *= (1 + porcentaje / 100);
        }
        printf("Precios aumentados en %.2f%%.\n", porcentaje);
        break;
    case 2:
             porcentaje = validarnumeros("Ingrese el porcentaje de reducción de precios: ", 0, PORCENTAJE_MAXIMO);
        for (int i = 0; i < cantidadElementos; i++)
        {
            inventario[i].precio *= (1 - porcentaje / 100);
        }
        printf("Precios reducidos en %.2f%%.\n", porcentaje);
        break;
    case 3:
              porcentaje = validarnumeros("Ingrese el porcentaje de aumento de cantidades: ", 0, PORCENTAJE_MAXIMO);
        for (int i = 0; i < cantidadElementos; i++)
        {
            inventario[i].cantidad *= (1 + porcentaje / 100);
        }
        printf("Cantidades aumentadas en %.2f%%.\n", porcentaje);
        break;
    case 4:
             porcentaje = validarnumeros("Ingrese el porcentaje de reducción de cantidades: ", 0, PORCENTAJE_MAXIMO);
        for (int i = 0; i < cantidadElementos; i++)
        {
            inventario[i].cantidad *= (1 - porcentaje / 100);
        }
```

```
        printf("Cantidades reducidas en %.2f%%.\n", porcentaje);
        break;
    }
}
```