Universidad Continental

Estructura de Datos

Docente: Carlos Enrique Del Carpio Rojas

NRC: 70715

Sistema de gestion de tienda de abarrotes “nombre”

INTEGRANTES:

- Llano Coaguila, Anthony Bryan
- Ccoropuna Alejandro, Celso Gustavo
- Jara Machicao, Alim Erasmo
- Cueva Alanguia, Cristian
- Jose Carlos Rivero Mamani
- Leovigildo Domingo Caya Umiyauri

PERÚ
2025

CAPÍTULO 1: Análisis del Problema

1. **Descripción del problema**
   El presente proyecto tiene como propósito desarrollar un sistema que permita gestionar de manera eficiente los productos de una tienda de abarrotes.

Se busca implementar un sistema sencillo en C++ que ayude a monitorear la información de los productos, facilitando el control de su nombre, precio y cantidad disponible. Este sistema permitirá realizar operaciones básicas como agregar, mostrar y eliminar productos, además de contar con un menú de navegación que mejore la interacción con el usuario.

El sistema está dirigido a la propia empresa y será utilizado por el personal encargado del área de almacén o administración, buscando optimizar sus tareas diarias mediante una herramienta confiable y fácil de usar.

2. Requerimientos del sistema

· Funcionales:

- Agregar producto: registrar un nuevo producto con sus respectivos datos (nombre, precio y cantidad).
- Mostrar productos: visualizar la lista de productos registrados con su información completa.
- Eliminar producto: quitar un producto del inventario.
- Menú principal: ofrecer opciones claras para acceder a las funciones del sistema.

· No funcionales:

- Los datos se almacenan de forma temporal (en memoria), no persistente.

3. Estructuras de datos propuestas

Se emplean estructuras tipo struct para definir las entidades principales del sistema:

Usuario: Contiene los campos nombre, apellido, edad y un puntero siguiente para enlazar varios usuarios.

```
struct usuario {

   string nombre;

   string apellido;

   int edad;

   usuario* siguiente;

};
```

Producto: Contiene los campos nombre, precio, cantidad y un puntero siguiente para formar una lista enlazada de productos.

```
struct productos {

   string nom_producto;

   double precio_producto;

   int cantidad_producto;

   productos* siguiente_pro;

};
```

Ambas estructuras forman listas enlazadas simples, donde cada nodo apunta al siguiente elemento. Esto permite agregar, recorrer y eliminar elementos dinámicamente sin usar arreglos de tamaño fijo.

4. **Justificación de la elección**
   Las listas enlazadas fueron seleccionadas porque:

Permiten almacenar un número variable de elementos sin definir previamente un tamaño máximo.

Facilitan la inserción y eliminación dinámica de nodos sin necesidad de desplazar los elementos.

Permiten recorrer los registros fácilmente para mostrarlos o realizar búsquedas simples.

Capítulo 2: Diseño de la Solución

1. **Descripción de estructuras de datos y operaciones:**

Estructuras de datos:

usuario → para registrar la información de quienes utilizan el sistema.

productos → para registrar, mostrar y eliminar productos.

Operaciones principales:

agregarProducto() → agrega un nuevo nodo al final de la lista de productos.

eliminar() → elimina un producto específico por su nombre.

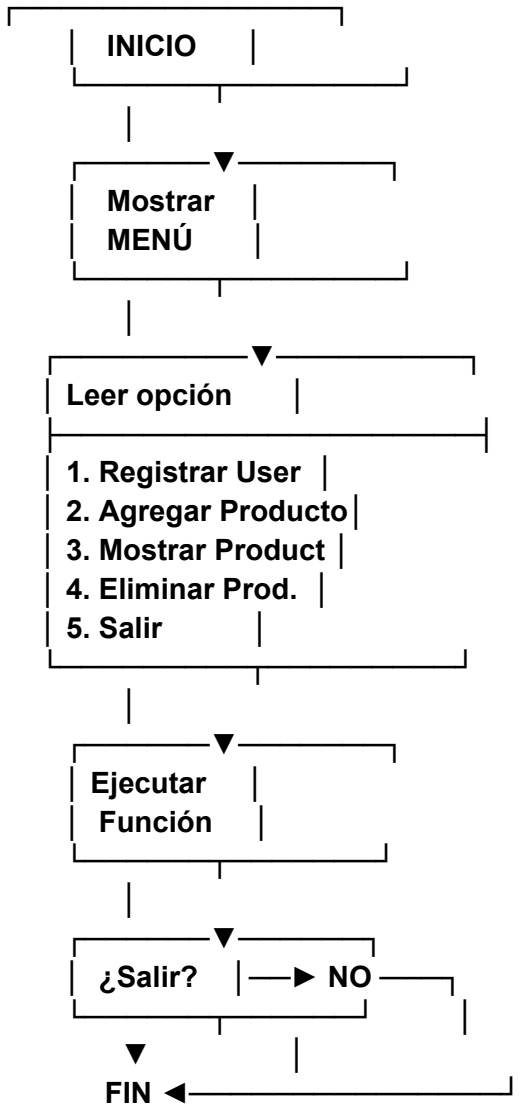
Menu() → muestra las opciones del sistema.

Uso de punteros (* y &) para recorrer, agregar o eliminar nodos dinámicamente

2. **Algoritmos principales:**

- *Pseudocódigo para agregar proceso.*
- *Pseudocódigo para cambiar el estado del proceso.*

3. **Diagramas de Flujo**

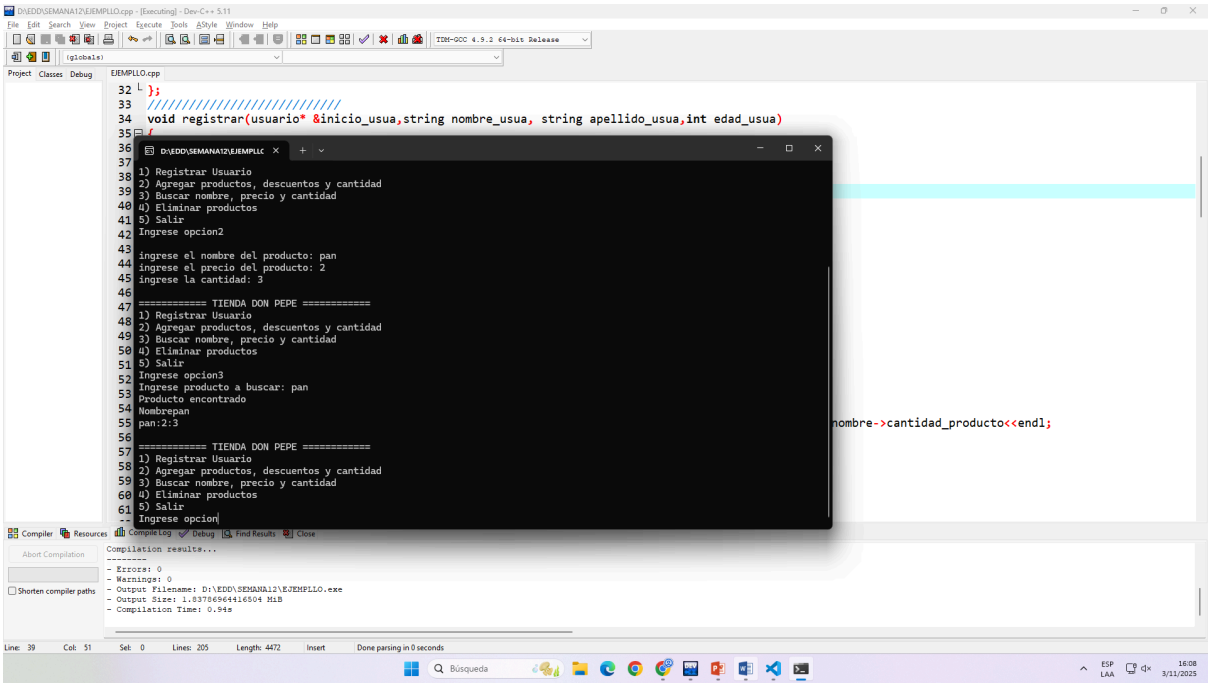


4. **Justificación del diseño:**
   (Ventajas, eficiencia, etc.)

Capítulo 3: Solución Final

1. **Código limpio, bien comentado y estructurado.**

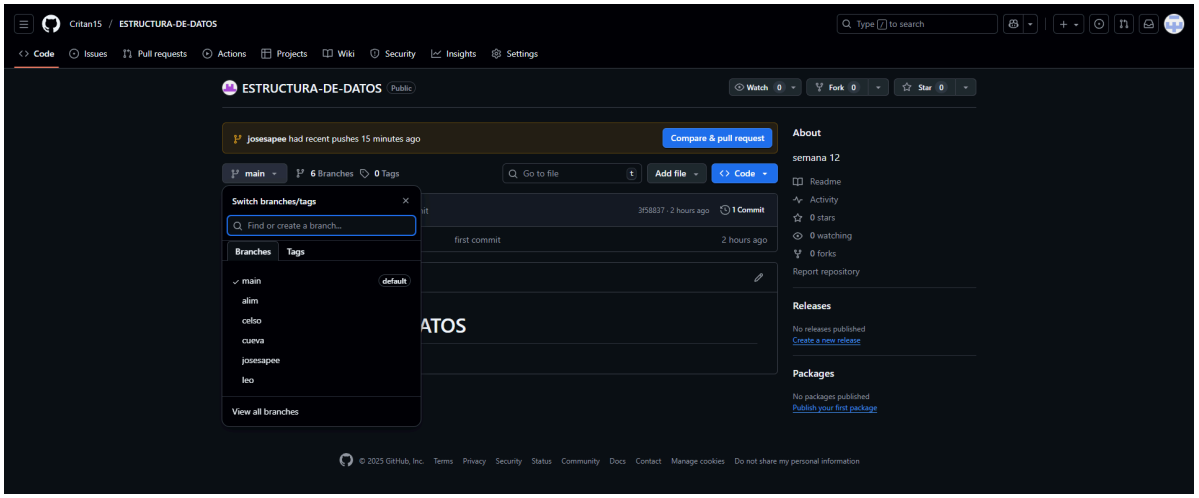
2. **Capturas de pantalla de las ventanas de ejecución con las diversas pruebas de validación de datos**



3. **Manual de usuario**

Capítulo 4: Evidencias de Trabajo en Equipo

1. **Repositorio con Control de Versiones (Capturas de Pantalla)**



- Registro de commits claros y significativos que evidencian aportes individuales (proactividad).
- Historial de ramas y fusiones si es aplicable.
- Evidencia por cada integrante del equipo.

- Enlace a la herramienta colaborativa

2. Plan de Trabajo y Roles Asignados

· Documento inicial donde se asignan tareas y responsabilidades.

- Llano Coaguila, Anthony Bryan -Avance del informe
- Ccoropuna Alejandro, Celso Gustavo -Avance del informe
- Jara Machicao, Alim Erasmo -Funciones menú y structs
- Cueva Alanguia, Cristian -Funciones agregar, eliminar
- Jose Carlos Rivero Mamani -Función mostrar todos los productos
- Leovigildo Domingo Caya Umiyauri -Función Buscar

.

Cronograma con fechas límite para cada entrega parcial.

Reunion: 3:20pm

Aprender GitHub: 3:30pm

Primer avance: 3:40pm

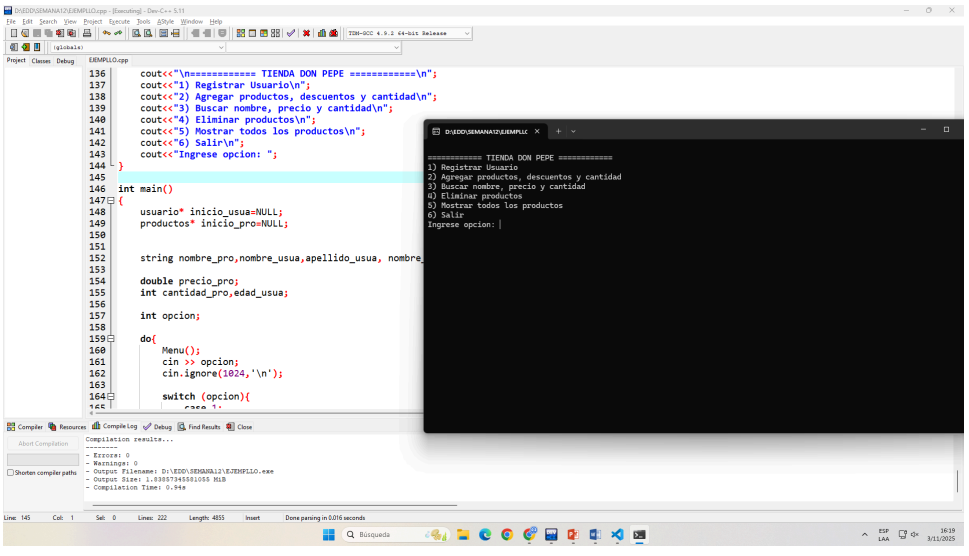
Segundo avance3:50pm

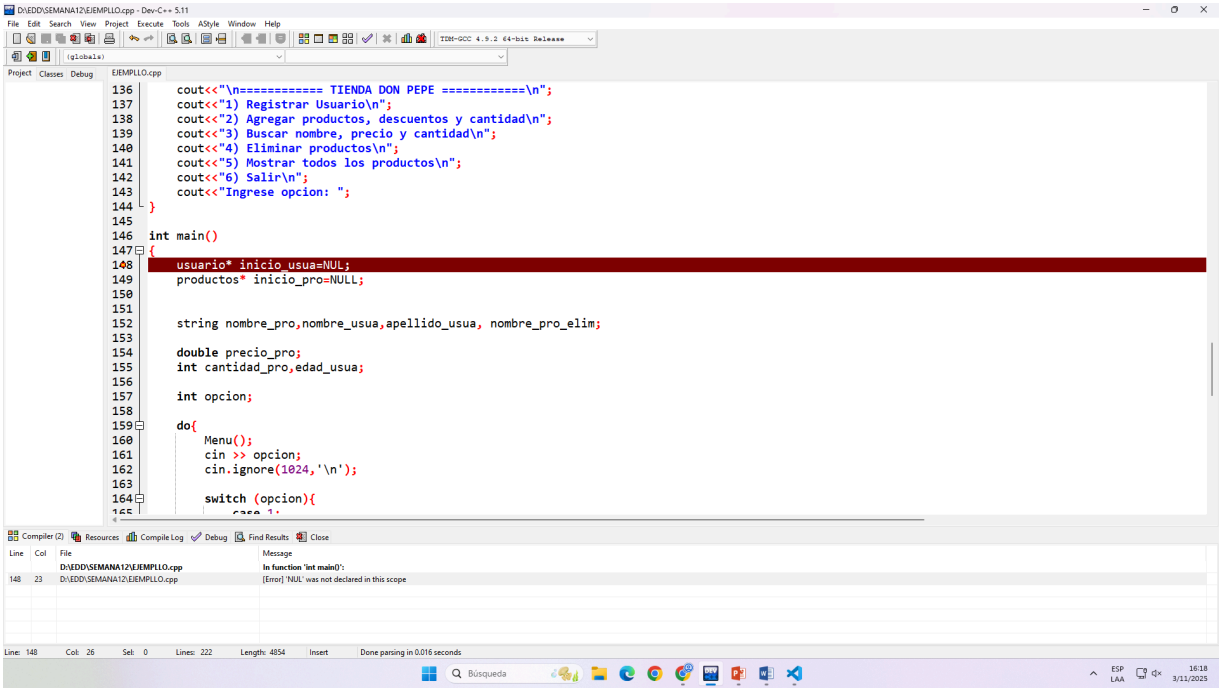
Entrega final: 4pm

Entrega FINAL FINAL: 4:30pm

· Registro de reuniones o comunicación del equipo (Actas de reuniones.).

0




D:\EDD\SEMANA12\EJEMPLLO.cpp - Dev-C++ 5.11
File Edit Search View Project Execute Tools AStyle Window Help
TDM-GCC 4.9.2 64-bit Release
(globals)
Project Classes Debug
EJEMPLLO.cpp
136 cout<<"\n============ TIENDA DON PEPE ============\n";
137 cout<<"1) Registrar Usuario\n";
138 cout<<"2) Agregar productos, descuentos y cantidad\n";
139 cout<<"3) Buscar nombre, precio y cantidad\n";
140 cout<<"4) Eliminar productos\n";
141 cout<<"5) Mostrar todos los productos\n";
142 cout<<"6) Salir\n";
143 cout<<"Ingrese opcion: ";
144 }
145
146 int main()
147 {
148 usuario* inicio_usua=NUL;
149 productos* inicio_pro=NULL;
150
151
152 string nombre_pro,nombre_usua,apellido_usua, nombre_pro_elim;
153
154 double precio_pro;
155 int cantidad_pro,edad_usua;
156
157 int opcion;
158
159 do{
160 Menu();
161 cin >> opcion;
162 cin.ignore(1024,'\n');
163
164 switch (opcion){
Compiler (2) Resources Compile Log Debug Find Results Close
Line Col File Message
D:\EDD\SEMANA12\EJEMPLLO.cpp In function 'int main()':
148 23 D:\EDD\SEMANA12\EJEMPLLO.cpp [Error] 'NUL' was not declared in this scope
Line: 148 Col: 26 Sel: 0 Lines: 222 Length: 4854 Insert Done parsing in 0.016 seconds
Búsqueda
ESP
LAA
16:18
3/11/2025