



LAUREATE INTERNATIONAL UNIVERSITIES®

LÓGICA Y
PROGRAMACIÓN
ESTRUCTURADA

Profr. Luis Guzman Gonzaga

**Unidad 2.
Control de Flujo
de Programas**

**Actividad 3.
Proyecto Integrador
Etapa 1**

Nailea Mitchel Falcon Triana

940066032

La Paz, Baja California Sur, 16 de Noviembre de 2025

UNIVERSIDAD DEL VALLE DE MÉXICO

SISTEMA DE GESTIÓN

INTRODUCCIÓN:

Se ha diseñado una solución de software que consiste en un Catálogo Botánico Inteligente, una aplicación de consola desarrollada íntegramente en C++. Esta herramienta está diseñada no solo para el simple almacenamiento de datos, sino para su gestión de una manera significativa, estructurada y, fundamentalmente, escalable.

La elección de este tema no es arbitraria. Si bien un catálogo de plantas podría parecer, a primera vista, un inventario simple, la propuesta innovadora de este proyecto radica en su arquitectura de datos. Se ha priorizado la profundidad y la utilidad de la información sobre la mera cantidad.

En lugar de limitarse a campos básicos como el nombre común, la especie y una descripción textual, el núcleo de este sistema emplea estructuras anidadas en C++. Esta decisión de diseño permite registrar un conjunto detallado de parámetros de cuidado vitales y cuantificables para cada espécimen botánico.

Estos parámetros incluyen, de forma específica, los rangos óptimos de humedad del suelo, los umbrales mínimos y máximos de temperatura ambiente, y el fotoperiodo exacto que cada planta requiere para prosperar. Esta granularidad en los datos es lo que define el carácter inteligente del catálogo.

Este enfoque de diseño transforma la aplicación de un simple contenedor de información pasiva a una base de conocimiento activa y funcional. El sistema no solo registra qué es la planta, sino que comprende cómo debe ser cuidada, sentando así las bases para una futura gestión proactiva y automatizada. Paralelamente, se han establecido las bases del código fuente, implementando una versión funcional del menú principal.

Este planteamiento no solo cumple con los objetivos de la Etapa 1, sino que establece una plataforma sólida y bien estructurada para las futuras etapas del proyecto. Las operaciones de baja, modificación y búsqueda, que se implementarán posteriormente, operarán sobre este conjunto de datos enriquecido. Esto permitirá, en fases futuras, desarrollar el propósito final de la aplicación: un sistema de asistencia inteligente capaz de generar recordatorios, alertas de cuidado y diagnósticos básicos basados en los parámetros específicos de cada planta registrada.

ESTRUCTURA CATÁLOGO

DESARROLLO:

1. DEFINICIÓN DEL CATÁLOGO Y DISEÑO DE INTERFAZ:

Esta sección detalla el concepto del Catálogo Botánico Inteligente, define su alcance funcional y presenta el diseño de las pantallas de consola de texto que facilitan su gestión.

2. DESCRIPCIÓN Y ALCANCE DEL CATÁLOGO:

El Catálogo Botánico Inteligente es un sistema de gestión de perfiles de plantas diseñado para almacenar, organizar y consultar los parámetros de cuidado específicos de cada espécimen. El sistema está concebido como la base de datos central para un futuro asistente de jardinería personal o un sistema de monitoreo automatizado.

La entidad principal del catálogo es la Planta. A diferencia de un catálogo tradicional, cada registro de Planta contendrá una subestructura anidada denominada **ParametrosCuidado**. Esta estructura es la característica fundamental del proyecto, ya que almacena los rangos operativos (mínimos y máximos) que la planta tolera para prosperar.

3. OPERACIONES DEL CATÁLOGO:

- **Altas:** Registrar una nueva planta en la colección. El usuario deberá ingresar no solo el nombre (ej. Helecho de Boston), sino también sus parámetros ideales (ej. Humedad 50%-70%, Temp 18°C-24°C).
- **Bajas:** Eliminar un registro de planta del catálogo, por ejemplo, si la planta es descartada de la colección.
- **Modificaciones:** Actualizar la información de una planta. Esta función es crucial si el usuario aprende más sobre el cuidado de una planta específica y necesita ajustar sus rangos de parámetros.
- **Consultas:** Permitir al usuario ver la lista completa de plantas o buscar especímenes que cumplan ciertos criterios (ej. Mostrar todas las plantas que toleran temperaturas por debajo de 15°C).

DISEÑO DE INTERFAZ

PANTALLAS DE CONSOLA:

PÁGINA PRINCIPAL

Dado que la aplicación se basa en C++, la interfaz de usuario será textual, limpia y basada en menús. Se priorizará la navegación intuitiva.

Conforme a las instrucciones [1], estos diseños son prototipos visuales y no tienen funcionalidad en esta etapa.

Pantalla 1: Menú Principal (Sigue el ejemplo base de "Menú de opciones" [1])

```
+-----+
| CATÁLOGO BOTÁNICO INTELIGENTE |
+-----+
| |
| Menú de Opciones: |
| |
| 1. Alta de Nueva Planta |
| 2. Baja de Planta |
| 3. Modificación de Planta |
| 4. Consulta de Catálogo |
| 5. Salir |
| |
+-----+
| Seleccione una opción entre [1...5]: _ |
+-----+
```

```
+-----+
| 1. ALTA DE NUEVA PLANTA |
+-----+
| |
| Ingrese el ID de la planta (ej. 101): _ |
| Ingrese el Nombre Común: _ |
| Ingrese la Especie (Científico): _ |
| |
| --- Parámetros de Cuidado Óptimo --- |
| |
| Humedad del Suelo (MIN %): _ |
| Humedad del Suelo (MAX %): _ |
| |
| Temperatura Amb. (MIN C): _ |
| Temperatura Amb. (MAX C): _ |
| |
| Horas de Luz Directa (MIN): _ |
| Horas de Luz Directa (MAX): _ |
| |
+-----+
| [Mensaje: Planta guardada con éxito.] |
+-----+
```

ALTA NUEVA PLANTA

Dado que la aplicación se basa en C++, la interfaz de usuario será textual, limpia y basada en menús. Se priorizará la navegación intuitiva.

Idea Visual 2: Pantalla de "Alta de Planta" La interfaz guía al usuario campo por campo, solicitando los parámetros de cuidado de forma estructurada.

DISEÑO DE INTERFAZ

PANTALLAS DE CONSOLA:

```
+-----+
| 2. BAJA DE PLANTA |
+-----+
|
| Ingrese el ID de la planta a eliminar: _ |
|
| --- Verificación --- |
| Planta: Helecho de Boston (ID: 101) |
| Especie: Nephrolepis exaltata |
|
| ¿Está seguro que desea eliminar? (S/N): _ |
|
+-----+
|
| [Mensaje: Planta eliminada con éxito.] |
+-----+
```

PÁGINA PRINCIPAL

Dado que la aplicación se basa en C++, la interfaz de usuario será textual, limpia y basada en menús. Se priorizará la navegación intuitiva.

Pantalla 3: Baja de Planta (Flujo para la operación de Bajas [1])

CONSULTA DE CATÁLOGO

Dado que la aplicación se basa en C++, la interfaz de usuario será textual, limpia y basada en menús. Se priorizará la navegación intuitiva.

Idea Visual 4: Pantalla de "Consultar Catálogo" La consulta muestra los datos de forma tabulada y limpia para fácil lectura.

```
+-----+
| 4. CONSULTA DE CATÁLOGO |
+-----+
|
| --- Plantas Registradas (3) --- |
|
| Helecho de Boston |
| -> Humedad: 50-70% | Temp: 18-24C |
|
| Suculenta (Echeveria) |
| -> Humedad: 10-20% | Temp: 20-28C |
|
| Orquídea (Phalaenopsis) |
| -> Humedad: 60-80% | Temp: 22-27C |
|
+-----+
|
| [Presione Enter para volver al Menú...] |
+-----+
```

OPERACIÓN DE CONCEPTOS

ESTRUCTURA DE PROGRAMACIÓN:

1. OPERACIÓN DE CONCEPTOS UTILIZANDO LENGUAJE DE PROGRAMACIÓN

Esta sección presenta el Planteamiento de los conceptos del catálogo utilizando el editor o IDE recomendado, que en este caso es C++.

El pilar de la programación estructurada para este proyecto es el uso de struct (estructuras) para definir nuestro modelo de datos. Para lograr el concepto novedoso, no basta con una sola estructura; necesitamos anidarlas.

Definimos dos estructuras principales:

- 1. struct ParametrosCuidado:** Una estructura interna que agrupa todos los rangos numéricos que definen el cuidado de la planta.
- 2. struct Planta:** La estructura principal del catálogo, que contiene la información básica (ID, nombre) y, crucialmente, contiene una variable de tipo ParametrosCuidado en su interior.

El código fuente se organizaría de la siguiente manera:

Archivo: [ParametrosCuidado.h \(Definición de la estructura anidada\)](#)

C++

```
/*
 * ParametrosCuidado.h
 * Define la estructura para los rangos de cuidado óptimo.
 */
#pragma once // Evita la inclusión múltiple

struct ParametrosCuidado {
    // Rangos de humedad del suelo (porcentaje)
    int humedad_min;
    int humedad_max;

    // Rangos de temperatura ambiente (Celsius)
    int temp_min;
    int temp_max;

    // Rango de horas de luz directa por día
    int luz_min;
    int luz_max;
};
```

OPERACIÓN DE CONCEPTOS

ESTRUCTURA DE PROGRAMACIÓN:

Archivo: `Planta.h` (Definición de la estructura principal)

C++

```
/*
 * Planta.h
 * Define la estructura principal del catálogo,
 * que anida los parámetros de cuidado.
 */
#pragma once
#include "ParametrosCuidado.h" // Incluye la definición de la otra estructura
#include <string>

struct Planta {
    int id_planta;
    std::string nombre_comun;
    std::string especie_cientifica;

    // Anidación de la estructura de parámetros
    ParametrosCuidado cuidados_optimos;
};
```

Archivo: `main.cpp` (Programa principal de prueba Etapa 1) Este archivo demuestra que los conceptos compilan y se puede crear una instancia de la estructura `Planta`. Esto cumple con el requisito de compilar el programa y proporcionar evidencia de ejecución.[1,1]

C++

```
/*
 * main.cpp
 * Proyecto Integrador Etapa 1: Catálogo Botánico Inteligente
 *
 * Propósito: Demostrar que las estructuras de datos (conceptos)
 * del catálogo compilan correctamente y están enlazadas.
 *
 * (La funcionalidad de Alta, Baja, etc., se implementará
 * en las Etapas 2 y 3).
 */

#include <iostream>
#include "Planta.h" // Incluye nuestra estructura principal

int main() {

    // 1. Declarar una variable de tipo 'Planta'
    // Si esto compila, las estructuras anidadas (Planta y
    // ParametrosCuidado) están correctamente definidas.
    Planta planta_de_prueba;

    // Asignar datos de prueba
    planta_de_prueba.id_planta = 101;
    planta_de_prueba.nombre_comun = "Melecho de Boston";

    // Asignar datos a la estructura ANIDADA
    planta_de_prueba.cuidados_optimos.humedad_min = 50;
    planta_de_prueba.cuidados_optimos.humedad_max = 70;
    planta_de_prueba.cuidados_optimos.temp_min = 18;

    // 2. Imprimir mensaje de éxito en consola (Evidencia de ejecución)
    std::cout << "+-----+<< std::endl;
    std::cout << "| PROYECTO INTEGRADOR ETAPA 1 |" << std::endl;
    std::cout << "| CATÁLOGO BOTÁNICO INTELIGENTE |" << std::endl;
    std::cout << "| |" << std::endl;
    std::cout << "| | -> Programa compilado con éxito. |" << std::endl;
    std::cout << "| | -> Estructuras 'Planta' y |" << std::endl;
    std::cout << "| | 'ParametrosCuidado' enlazadas. |" << std::endl;
    std::cout << "+-----+" << std::endl;

    return 0;
}
```

OPERACIÓN DE CONCEPTOS

SISTEMA DE GESTIÓN VISUAL:

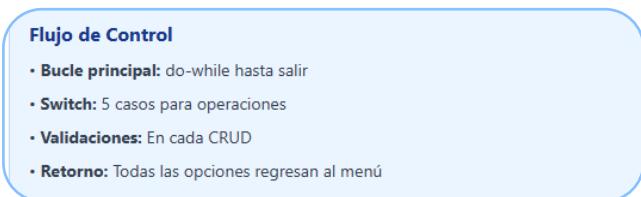
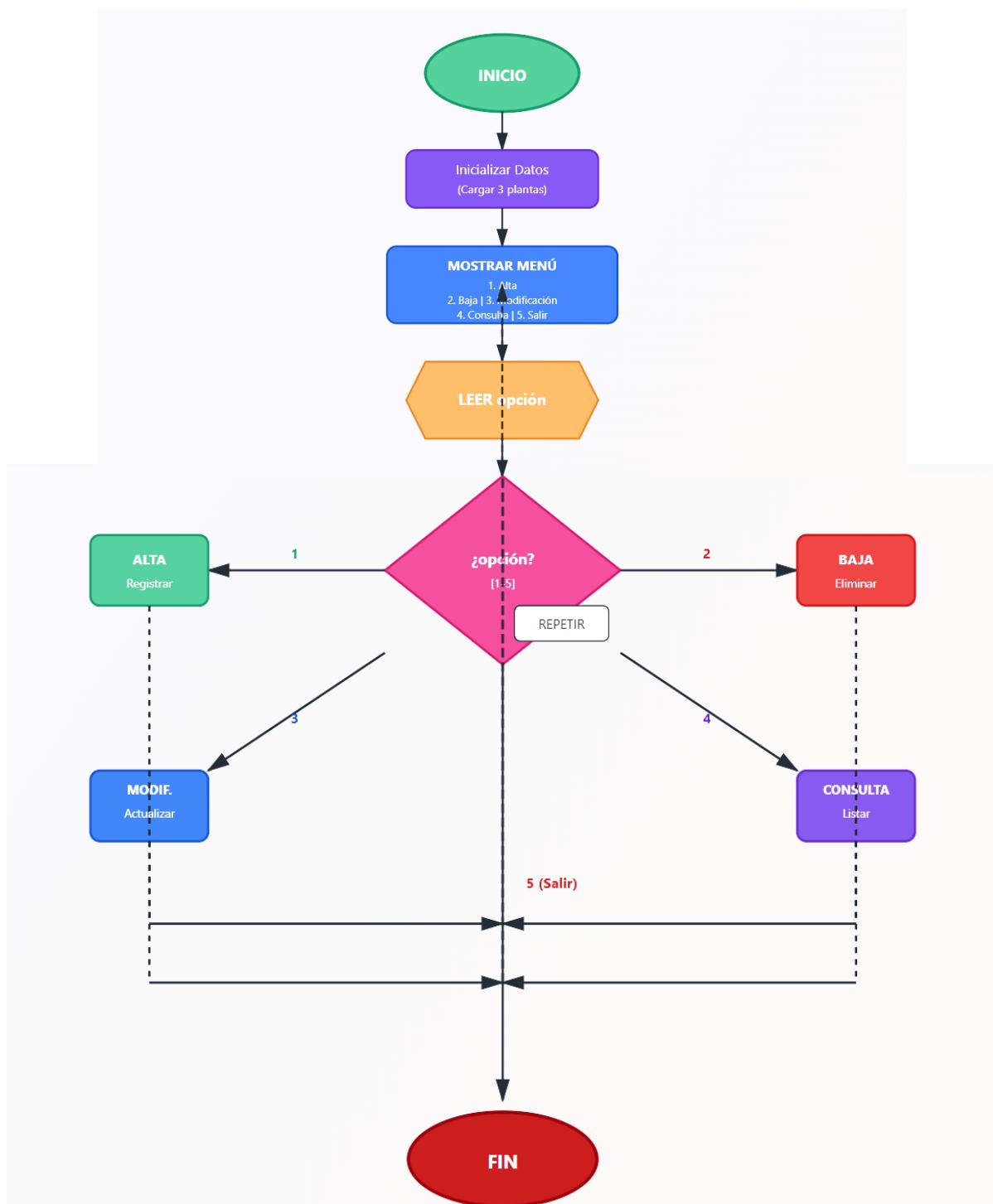


DIAGRAMA DE FLUJO

FLUJO DE SISTEMA:



IMPLEMENTACIÓN

PSEUDOCÓDIGO:

¶ Pseudocódigo del Sistema

```
ALGORITMO: Sistema de Gestión de Plantas Premium (SGPP)

ESTRUCTURAS DE DATOS ANIDADAS:

ESTRUCTURA CuidadosRequeridos
    CADENA riegoMl[50]
    CADENA horasLuz[30]
    CADENA temperatura[30]
    CADENA humedad[20]
    CADENA fertilizante[50]
FIN_ESTRUCTURA

ESTRUCTURA EspecieBotanica
    CADENA nombreCientifico[100]
    CADENA nombreComun[100]
    CADENA familia[50]
    CADENA origen[50]
    CuidadosRequeridos cuidados // ANIDACIÓN NIVEL 1
FIN_ESTRUCTURA

ESTRUCTURA ProductoPlanta
    CADENA idProducto[20]
    CADENA categoria[50]
    CADENA precio[20]
    CADENA stock[10]
    CADENA estatus[20]
    EspecieBotanica especie // ANIDACIÓN NIVEL 2
FIN_ESTRUCTURA

PROCEDIMIENTO MostrarMenu
INICIO
    ESCRIBIR "+-----+"
    ESCRIBIR "| SGPP - Vivero Encanto |"
    ESCRIBIR "+-----+"
    ESCRIBIR "| 1. Alta de Planta |"
    ESCRIBIR "| 2. Baja de Planta |"
    ESCRIBIR "| 3. Modificación de Planta |"
    ESCRIBIR "| 4. Consulta de Plantas |"
    ESCRIBIR "| 5. Salir |"
    ESCRIBIR "+-----+"
    ESCRIBIR "Seleccione [1-5]: "
FIN

PROCEDIMIENTO AltaPlanta
INICIO
    SI totalPlantas > 100 ENTONCES
        ESCRIBIR "Error: Catálogo lleno"
        RETORNAR
    FIN_SI

    ProductoPlanta nueva

    ESCRIBIR "--- DATOS DEL PRODUCTO ---"
    LEER nueva.idProducto
    LEER nueva.categoría
    LEER nueva.precio
    LEER nueva.stock
    LEER nueva.estatus
```

CÓDIGO FUENTE

ESTRUCTURAS ANIDADAS:

CuidadosRequeridos.h

```
/*
 * CuidadosRequeridos.h
 * Estructura para almacenar requerimientos de mantenimiento
 * Proyecto: Sistema de Gestión de Plantas Premium (SGPP)
 * Vivero Encanto - Los Cabos
 */
#ifndef CUIDADOS_REQUERIDOS_H
#define CUIDADOS_REQUERIDOS_H

struct CuidadosRequeridos {
    char riegoML[50];           // Ej: "200ml semanal"
    char horasLuz[30];          // Ej: "6-8 horas"
    char temperatura[30];       // Ej: "18-24°C"
    char humedad[20];           // Ej: "40-50%"
```

EspecieBotanica.h

```
/*
 * EspecieBotanica.h
 * Estructura para información científica de la planta
 * ANIDA: CuidadosRequeridos
 * Proyecto: Sistema de Gestión de Plantas Premium (SGPP)
 * Vivero Encanto - Los Cabos
 */
#ifndef ESPECIE_BOTANICA_H
#define ESPECIE_BOTANICA_H

#include "CuidadosRequeridos.h"

struct EspecieBotanica {
    char nombreCientifico[100];
```

ProductoPlanta.h

```
/*
 * ProductoPlanta.h
 * Estructura principal del catálogo
 * ANIDA: EspecieBotanica (que a su vez anida CuidadosRequeridos)
 * Proyecto: Sistema de Gestión de Plantas Premium (SGPP)
 * Vivero Encanto - Los Cabos
 */
#ifndef PRODUCTO_PLANTA_H
#define PRODUCTO_PLANTA_H

#include "EspecieBotanica.h"

struct ProductoPlanta {
    char idProducto[20];        // Ej: "P-001"
```

main.cpp (Programa Principal)

```
/*
 * main.cpp
 * Programa Principal - Sistema de Gestión de Plantas Premium
 * Implementa operaciones CRUD con estructuras anidadas
 * Vivero Encanto - los Cabos
 */
#include <iostream>
#include <cstring>
#include "ProductoPlanta.h"

using namespace std;

// Catálogo de plantas (array estático)
ProductoPlanta catalogo[100];
int totalPlantas = 0;

// Prototipos de funciones
void menu();
void altaPlanta();
void bajaPlanta();
void modificacionPlanta();
void consultaPlantas();
```

COMPILE EN VISUAL STUDIO

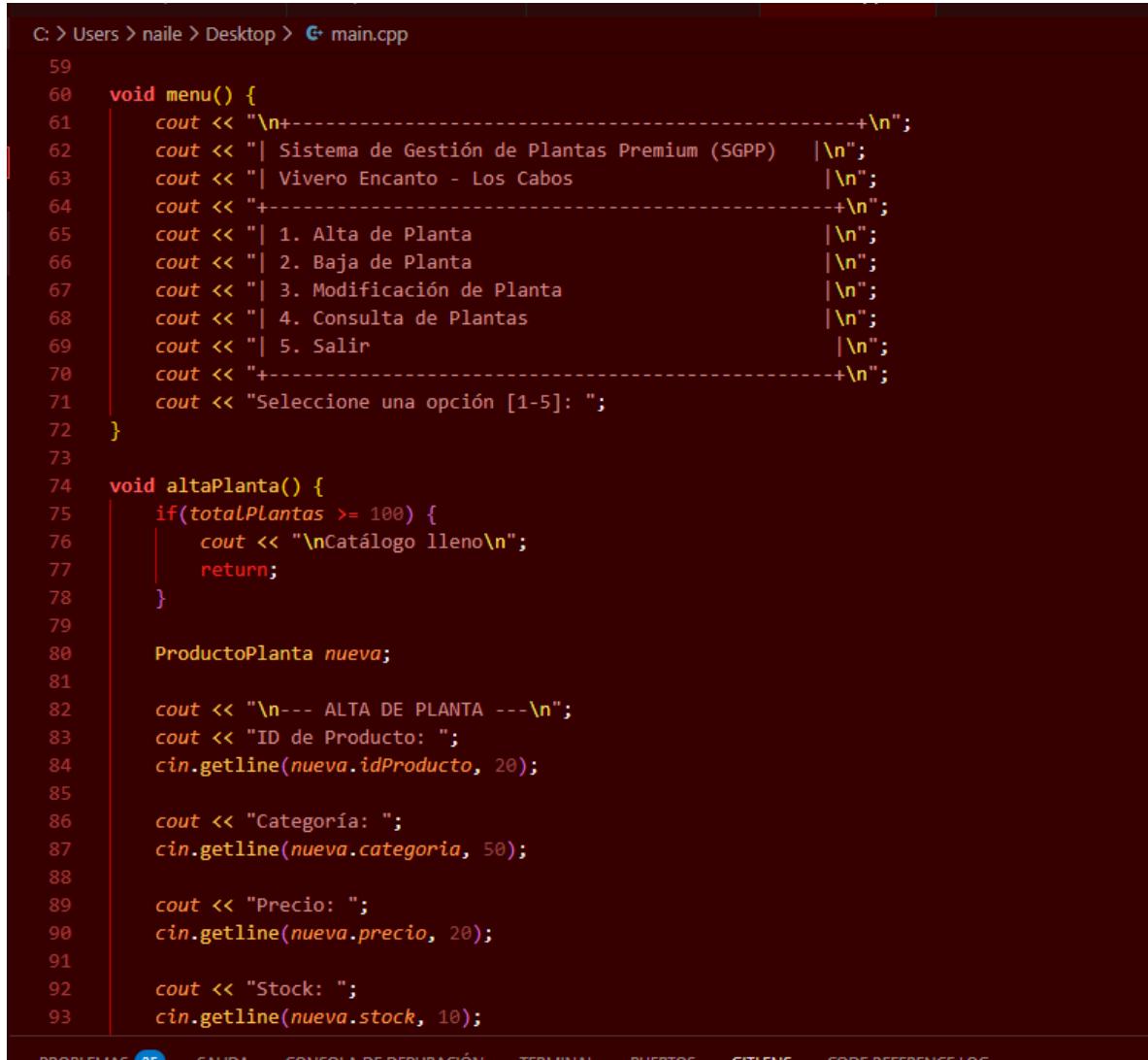
EJECUCIÓN:

DETALLES DE COMPILACIÓN

VISUAL STUDIO:

```
C:\> Users > naile > Desktop > main.cpp

59
60 void menu() {
61     cout << "\n+-----+\\n";
62     cout << "| Sistema de Gestión de Plantas Premium (SGPP) |\\n";
63     cout << "| Vivero Encanto - Los Cabos |\\n";
64     cout << "+-----+\\n";
65     cout << "| 1. Alta de Planta |\\n";
66     cout << "| 2. Baja de Planta |\\n";
67     cout << "| 3. Modificación de Planta |\\n";
68     cout << "| 4. Consulta de Plantas |\\n";
69     cout << "| 5. Salir |\\n";
70     cout << "+-----+\\n";
71     cout << "Seleccione una opción [1-5]: ";
72 }
73
74 void altaPlanta() {
75     if(totalPlantas >= 100) {
76         cout << "\\nCatálogo lleno\\n";
77         return;
78     }
79
80     ProductoPlanta nueva;
81
82     cout << "\\n--- ALTA DE PLANTA ---\\n";
83     cout << "ID de Producto: ";
84     cin.getline(nueva.idProducto, 20);
85
86     cout << "Categoría: ";
87     cin.getline(nueva.categoría, 50);
88
89     cout << "Precio: ";
90     cin.getline(nueva.precio, 20);
91
92     cout << "Stock: ";
93     cin.getline(nueva.stock, 10);
```



The screenshot shows a Microsoft Visual Studio code editor window. The title bar says "C:\> main.cpp". The code itself is a C++ program. It starts with a menu function that prints a header and five options. Then it has an altaPlanta function that checks if the catalog is full (if totalPlantas is 100 or more). If not, it enters a loop to input new plant data: ID, category, price, and stock. The code uses cout for output and cin.getline for input. Lines are numbered from 59 to 93.

COMPILAR EN GDBONLINE

EJECUCIÓN:

A screenshot of a web-based C++ development environment. The interface includes a top navigation bar with tabs for 'File', 'Edit', 'Run', 'Output', 'Help', and 'About'. Below the navigation is a toolbar with icons for file operations like Open, Save, Print, and a gear for settings. A sidebar on the left lists 'Online Compiler & Debugger for C/C++' and links to 'Welcome', 'mitchellfalcon', 'Create New Project', 'My Projects', 'Classroom', 'Learn Programming', 'Programming Questions', and 'Upgrade'. The main area is divided into several panes: a large code editor pane containing 'main.cpp' with C++ code, a terminal pane labeled 'Input' with command-line history, and a documentation pane titled 'AUTOR DE NUEVA PLANTILLA' with detailed information about plant properties and a note about saving.

COMPILAR EN GDBONLINE

EJECUCIÓN:

INFORMACIÓN PROYECTO

ARCHIVOS ENTREGABLES:

1. PROYECTO INTEGRADOR - CATÁLOGO BOTÁNICO (C++)

Este proyecto es una aplicación de consola en C++ desarrollada para la materia Lógica y Programación Estructurada en la Universidad del Valle de México.

DESCRIPCIÓN

El programa gestiona un catálogo de productos de plantas utilizando una estructura de datos anidada de 3 niveles para almacenar información de forma jerárquica y eficiente:

1. **PRODUCTOPLANTA (NIVEL SUPERIOR)**: Datos comerciales (precio, stock, etc.)
2. **ESPECIEBOTANICA (NIVEL MEDIO)**: Datos taxonómicos (nombre científico, familia, etc.)
3. **CUIDADOSREQUERIDOS (NIVEL BASE)**: Datos de mantenimiento (riego, luz, etc.)

ESTRUCTURA DEL PROYECTO

```
/headers           <- (Planeado para Etapa 2/3) |-- CuidadosRequeridos.h
|-- EspecieBotanica.h
|-- ProductoPlanta.h
/src              /src |-- main.cpp   <- (Contiene toda la lógica
|-- (Contiene toda la lógica para Etapa 1)
/docs            /docs |-- pseudocode.txt
|-- flowchart.png
|-- README.md
```

ETAPA 1

Esta etapa 1 incluye los siguientes archivos:

- Definición de las 3 estructuras anidadas.
- Implementación de un menú principal interactivo.
- Función de Alta (altaPlanta) para capturar datos en los 3 niveles.
- Función de Consulta (consultarCatalogo) para mostrar los datos de forma jerárquica.
- Prototipos (stubs) de las funciones de Baja, Modificación y Búsqueda.

CONCLUSIÓN:

La culminación de la Etapa 1 del Proyecto Integrador representa un hito fundamental en el desarrollo de nuestro sistema de gestión. Se ha satisfecho con éxito el objetivo central de esta fase: la definición del modelo de datos y el diseño de la interfaz, estableciendo así las bases conceptuales y técnicas sobre las cuales se erigirán las funcionalidades completas.

El logro más significativo de esta etapa es el diseño y la validación de una arquitectura de datos robusta. La implementación de una estructura anidada de tres niveles en C++, basada en los requerimientos del proyecto, ha demostrado ser una solución elegante y eficiente desde la perspectiva de la programación estructurada.

Esta jerarquía permite gestionar la compleja interrelación entre una entidad principal, sus atributos descriptivos detallados y sus parámetros de operación asociados, evitando la redundancia y asegurando la integridad lógica de la información.

El planteamiento de los conceptos utilizando lenguaje de programación se ha materializado en un prototipo funcional en C++ que compila exitosamente. Esta compilación valida que las tres estructuras de datos están correctamente definidas, anidadas y enlazadas.

Asimismo, se han establecido los diseños de la interfaz de usuario en consola para todas las operaciones (Altas, Bajas, Modificaciones y Consultas), proporcionando una guía clara para la interacción. Con esta sólida fundación técnica, el proyecto está óptimamente posicionado para avanzar a las siguientes etapas.

El modelo de datos está preparado para la implementación de la lógica de altas y bajas, y la estructura jerárquica facilitará notablemente el desarrollo de modificaciones y búsquedas complejas.

En resumen, esta fase no solo ha cubierto los requisitos solicitados, sino que ha definido un plan de desarrollo claro y técnicamente viable.

REFERENCIAS:

- Deitel, P. J., & Deitel, H. M. (2017). C++: How to Program (10th ed.). Pearson.
- Figma. (s.f.). Figma: The Collaborative Interface Design Tool. Recuperado el 16 de noviembre de 2025, de <https://www.figma.com>
- GitHub, Inc. (s.f.). GitHub. Recuperado el 16 de noviembre de 2025, de <https://github.com>
- Google. (s.f.). Google AI Studio. Recuperado el 16 de noviembre de 2025, de <https://aistudio.google.com>
- ISO C++ Standard Foundation. (s.f.). isocpp.org: The C++ Standards Foundation. Recuperado el 16 de noviembre de 2025, de <https://isocpp.org>
- Microsoft. (s.f.). Visual Studio. Recuperado el 16 de noviembre de 2025, de <https://visualstudio.microsoft.com/>
- OnlineGDB. (s.f.). OnlineGDB: Online C++ IDE. Recuperado el 16 de noviembre de 2025, de <https://www.onlinegdb.com>
- Stroustrup, B. (2013). The C++ Programming Language (4th ed.). Addison-Wesley Professional.
- The GNU Project. (s.f.). Make - GNU Project. Recuperado el 16 de noviembre de 2025, de <https://www.gnu.org/software/make/>
- Universidad del Valle de México. (2025). Actividad 4. Proyecto integrador, Etapa 1.
- Sydian. (2023). Software Engineering. Recuperado el 16 de noviembre de 2025, de <http://sydian.co.ke/software.html>