CUCEI

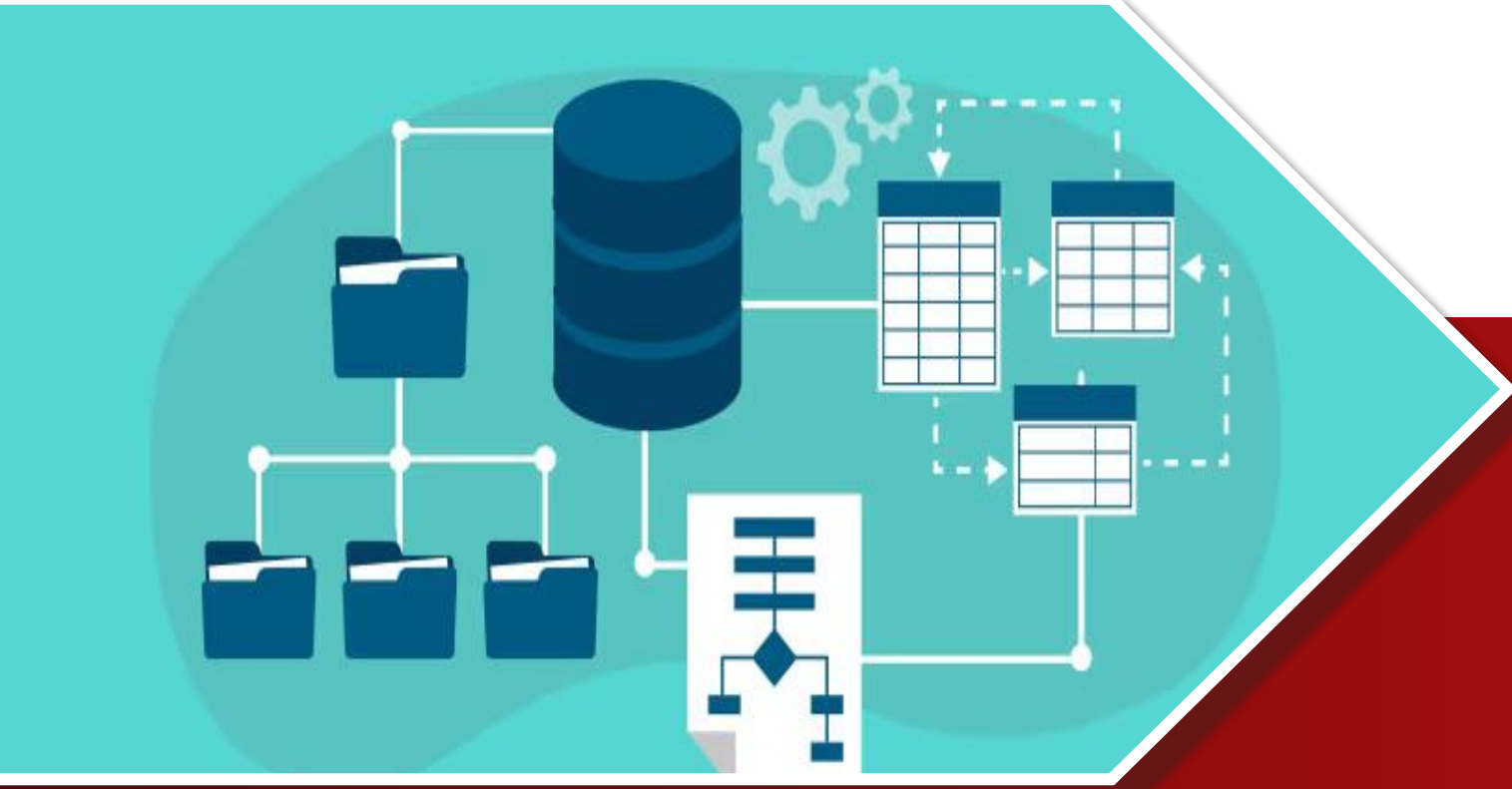
Actividad de Aprendizaje 02

La Anidación Estructural: Registros con Arreglos, Arreglos de Registros y Arreglos de Objetos

Centro Universitario de Ciencias Exactas e Ingenierías

Materia: Estructuras de Datos

Clave: V0731 ***Sección***: D02



Alumno: Mariscal Rodríguez Omar Jesús

Código: 220858478

Profesor: Gutiérrez Hernández Alfredo

Fecha: 24 de Agosto de 2025

Contenido

Test de Autoevaluación

Autoevaluación			
Concepto	***Sí***	***No***	***Acumulado***
Bajé el trabajo de internet o alguien me lo pasó (aunque sea de forma parcial)	***-100 pts***	***0 pts***	***0***
Incluí el código fuente ***en formato de texto (sólo si funciona cumpliendo todos los requerimientos)***	***+25 pts***	***0 pts***	***25***
Incluí las ***impresiones de pantalla (sólo si funciona cumpliendo todos los requerimientos)***	***+25 pts***	***0 pts***	***25***
Incluí una ***portada*** que identifica mi trabajo (nombre, código, materia, fecha, título)	***+25 pts***	***0 pts***	***25***
Incluí una ***descripción y conclusiones*** de mi trabajo	***+25 pts***	***0 pts***	***25***
		Suma:	***100***

Introducción y Abordaje del Problema

Esta actividad tuvo su objetivo en seguir siendo un repaso, ahora tratando estructuras de datos anidadas.

Podríamos dividir este abordaje de problema en dos: Planeación y Programación

Planeación

Para abordar esta actividad ***requirió de visualizar cómo se debía ver el resultado final***; comencé ***haciendo un boceto sobre papel de cómo deberían de quedar*** las distintas impresiones de pantallas. Primeramente, fue para ver ***qué flujo seguirían***, después ***añadirle un poco de estilo*** sobre cómo se verían mejor que solo texto plano.

Después de tener un bosquejo sobre todo coherente con lo que debería hacer el programa, pasé a planear las distintas clases que tendría el programa, ya se nos daba una propuesta: ***Fecha, Producto, Colección y Menú.*** Este fue el punto de partida, sin embargo, como lo hice con la actividad anterior, agregué primeramente la clase de "***utilities***" que sirve para la lectura de datos, aunque en este trabajo, le di un pequeño giro; aquí se pueden ingresar tanto cantidades enteras como reales, un ejemplo son las existencias y el precio, me convendría que "***utilities"*** pudiera manejar un ciclo en el que se saliera solo si se ingresan cantidades entre un rango. Al tener las posibilidades de dos tipos de datos, tanto int, como float, podría haber hecho funciones para cada una, pero recurrí a utilizar ***templates,*** de esta manera, una función que limite la entrada a un rango sirve para ambos datos; de pasó, utilizando la biblioteca ***limits,*** planee que estas funciones también reaccionaran si se ingresa otro tipo de dato, como una letra, los tendría que reconocer y ejecutar el bucle. Además, utilities deberá tener un método para solo ingresar cantidades positivas mayores a 0, lo pensé para datos como el precio o el peso, que no pueden ser negativos.

Cuando leí la actividad y que tendríamos que hacer un arreglo estático de objetos de al menos 500, pensé en tener un define para determinar la capacidad del arreglo, de esta manera sería muy fácil cambiar el tope del arreglo solo en un archivo; investigando por la web, para C++ y POO no recomiendan usar tanto define globales, una de las razones es porque los define no especifican el tipo de dato y eso puede llegar a tener problemas. Mi solución fue crear un nuevo archivo llamado configure.hpp donde tendría un namespace llamado configure y ahí están como variables inline y constantes, variables

clave como el tope estático del arreglo, el límite de caracteres del código de barras o el año mínimo en el cuál pueden ingresar la fecha de entrada. Todas estas variables que están presenten en distintas partes del código, podría controlarlas desde un solo lugar y solo modificando una línea.

Otra cosa clara que tenía es que el constructor del menú solo tendría el método privado del menú principal, donde se ejecutaría todos los demás submenús, dejando así, limpio el main a la espera de solo importar e instanciar la clase menú.

Tras este proceso, quedé que mi programa tendría las siguientes clases:

Date

Product

Utilities

Menu

Inventory

Nota: Configure a priori no es un objeto como tal, sino solo un namespace.

Programación

A la hora de programar, sabiendo qué clases y los métodos que tendrían fue bastante ágil, algo que señalaría es lo que pasó con Utilities. Esta clase es concebida como una clase estática (que no necesita de instanciarse para usar sus métodos) y ahora también maneja templates, sin embargo, tenía algunos problemas de compilación al separarlo en su encabezado y definición, investigando, tendría que usar otro tipo de archivo de extensión. tpp e indicarle al compilador, por lo que opté por dejar las definiciones en el mismo encabezado.

Los distintos setter's de las clases tienen validaciones internas, algo que no había hecho en mi trabajo anterior porque pensaba que bastaría con validaciones externas, pero aprendí que se necesitan también internas para favorecer el encapsulamiento, por lo que las clases como Date, tienen validaciones internas, que se ingresa un dato inválido, suelta una excepción acorde con un mensaje, en el caso de que se ingrese un año inválido se lanza un out_of_range, o en el setter para establecer la existencia actual de un producto

se lanza un invalid_argument, avisando al programador qué el objeto no puede establecer ese estado irreal.

A pesar de tener esas validaciones, el menú, encargado de la interacción con el usuario también maneja sus depuraciones y avisa si los datos son incorrectos para evitar situaciones donde el programa pueda funcionar de manera rara.

Con la planeación previamente hecha, la programación fue bastante más rápida, solo fue ir solucionando problemas y bugs que surgían por el camino.

Código Fuente

Carpeta include

configure.hpp

```
// Constantes de Configuración para poder Ajustar ciertos parámetros del
// programa con un sólo edit
#ifndef __CONFIGURE_H__
#define __CONFIGURE_H__

namespace configure {
inline constexpr int inventoryCapacity =
    500;  // Capacidad del arreglo estático del inventario
inline constexpr int barcodeSize =
    13;  // Tope de caracteres en el código de barras
inline constexpr int minimumYear =
    1900;  // Año mínimo para el entryDate de los Productos
}  // namespace configure

#endif  // __CONFIGURE_H__
```

date.hpp

```
#ifndef __DATE_H__
#define __DATE_H__

#include <chrono> //Constructor base inicializará los parámetros al día
de hoy
#include <iostream>

#include "configure.hpp"

class Date {
 private:
  int year;
  int month;
  int day;

 public:
  // Constructores

  Date();
  Date(const Date&);
  Date(const int&, const int&, const int&);

  // Interfaces
  // Getter's
  int getYear() const;
  int getMonth() const;
  int getDay() const;

  std::string toString() const;

  // Setter's
  void setYear(const int&);
  void setMonth(const int&);
  void setDay(const int&);

  Date& operator=(const Date&);
};

#endif  // __DATE_H__
```

inventory.hpp

```
#ifndef __INVENTORY_H__
#define __INVENTORY_H__

#include <iomanip>  //Para hacer más estéticos las impresionse de
pantalla
#include <string>

#include "configure.hpp"
#include "product.hpp"

class Inventory {
 private:
  int numberOfProducts;  // Número de elementos registrados
  Product
      products[configure::inventoryCapacity];  // Arreglo estático de
productos.
                                               // Se puede modificar
desde el
                                               // archivo configure.hpp

 public:
  // Constructores
  Inventory();
  Inventory(const Inventory&);
  Inventory(const int&, const Product*);

  // Interfaces:
  // Getter's
  int getNumberOfProducts() const;
  const Product* getProducts() const;

  std::string toString() const;

  // Setter's
  void setNumberOfProducts(const int&);
  void setProducts(const Product*);

  Inventory& operator=(const Inventory&);

  // Algoritmitos
  Product* searchProduct(
      const std::string&);  // Buscador de productos. Devuelve un
apuntador

  void addProduct(const Product&);  // Agregar un producto

  void increaseProduct(const std::string&,
```

```
                        const int&);  // Incrementa la cantidad en
existencia
                                      // dado un código de barras
  void decreaseProduct(const std::string&,
                        const int&);  // Decrece la cantidad en existencia
dado
                                      // un código de barras
};

#endif  // __INVENTORY_H__
```

menu.hpp

```
#ifndef __MENU_H__
#define __MENU_H__

#include <iomanip>  //Para hacer más estéticos las impresiones
#include <iostream>

#include "inventory.hpp"
#include "utilities.hpp"

class Menu {
 private:
  Inventory
      inventory;  // Tiene de parámetros un objeto de la clase
inventario. Al
                  // tenerlo de atributo, facilita el acceso a sus
métodos

  // Pantallas del Programa
  void mainMenu();
  void newProduct();
  void increaseStock();
  void decreaseStock();
  void checkInventory();
  void searchProduct();
  void exitScreen();

 public:
  Menu();
};

#endif  // __MENU_H__
```

product.hpp

```
#ifndef __PRODUCT_H__
#define __PRODUCT_H__

#include <iomanip>
#include <string>

#include "configure.hpp"
#include "date.hpp"

class Product {
 private:
  std::string barcode;      // Código de Barras
  std::string productName;  // Nombre del Producto
  float weight;             // Peso
  Date entryDate;           // Fecha de Entrada
  float wholesalePrice;     // Fecha de Mayoreo
  float retailPrice;        // Fecha de Menudeo
  int currentExistence;     // Existencia Actual
 public:
  // Constructores.
  // Al tener muchos atributos, se omitió el constructor paramétrico
  Product();
  Product(const Product&);

  // Interfaces
  // Getter's
  std::string getBarcode() const;
  std::string getProductName() const;
  float getWeight() const;
  Date getEntryDate() const;
  float getWholesalePrice() const;
  float getRetailPrice() const;
  int getCurrentExistence() const;

  std::string toString() const;

  // Setter's
  void setBarcode(const std::string&);
  void setProductName(const std::string&);
  void setWeight(const float&);
  void setEntryDate(const Date&);
  void setWholesalePrice(const float&);
  void setRetailPrice(const float&);
  void setCurrentExistence(const int&);

  Product& operator=(const Product&);
```

```
    // Algoritmicos
    void increaseExistence(const int&);  // Incrementar las existencias
    void decreaseExistence(const int&);  // Decrecer las existencias
};

#endif  // __PRODUCT_H__
```

utilities.hpp

```
/*utilities.hpp es una serie de acciones útiles para reciclar código.
Es una clase abstracta, no necesita instanciarse, y tiene métodos para
leer un
rango de números o leer números exclusivamente enteros > 0 Utiliza
templates, y
para tener la combinación de templates y métodos estáticos sin
complicarse con
el compilador, va en el mismo header. Otra opción para tener los archivos
serparados sería tener un .tpp o indicar la ruta al compilador
*/

#ifndef __UTILITIES_H__
#define __UTILITIES_H__

#include <iostream>
#include <limits>  //Para verificar que la entrada sea del tipo pedido
#include <string>

template <typename T>
class Utilities {
 public:
  // lectura de datos enteros o flotantes gracias al template
  static T readNumberRange(const T&, const T&, const std::string&);
  static T readPositiveNumber(const std::string&);
};

template <typename T>
T Utilities<T>::readNumberRange(const T& lowerLimit,
                                const T& upperLimit,
                                const std::string& prompt) {
  T value;

  // Iteración perpetua hasta que se introduzca un valor válido
  while (true) {
    std::cout << prompt;  // Mensaje
    std::cin >> value;    // Valor Resultado

    // Verificar si se ingresó algo diferente a un número
    if (std::cin.fail()) {
      std::cin.clear();  // Limpiar el error
      std::cin.ignore(std::numeric_limits<std::streamsize>::max(),
                      '\n');  // Vaciar el buffer
      system("CLS");
      std::cout << "Entrada invalida.\n Por favor, intetelo
nuevamente.\n\n";
      system("PAUSE");
      continue;  // Volver al inicio del loop
```

```
    }

    // Vaciar el buffer
    std::cin.ignore(std::numeric_limits<std::streamsize>::max(), '\n');

    // Verificar el rango
    if (value < lowerLimit || value > upperLimit) {
      system("CLS");
      std::cout
          << "Numero fuera de rango.\n Por favor, intetelo
nuevamente.\n\n";
      continue;  // Volver al inicio de la iteción
    }

    // Entrada correcta
    return value;
  }  // Bucle indefinido hasta que se dé un valor compelto
}

// Misma dinámica pero con valores exclusivamente > 0
template <typename T>
T Utilities<T>::readPositiveNumber(const std::string& prompt) {
  T value;

  while (true) {
    std::cout << prompt;
    std::cin >> value;

    // Verificar si se ingresó algo diferente a un número
    if (std::cin.fail()) {
      std::cin.clear();  // Limpiar el error
      std::cin.ignore(std::numeric_limits<std::streamsize>::max(),
                      '\n');  // Vaciar el buffer
      system("CLS");
      std::cout << "Entrada invalida.\n Por favor, intetelo
nuevamente.\n\n";
      system("PAUSE");
      continue;  // Volver al inicio del loop
    }

    // Vaciar el buffer
    std::cin.ignore(std::numeric_limits<std::streamsize>::max(), '\n');

    // Verificar el rango
    if (value <= 0) {
      system("CLS");
      std::cout
```

```
            << "Numero fuera de rango.\n Por favor, intetelo
nuevamente.\n\n";
        continue;  // Volver al inicio de la iteción
    }

    // Entrada correcta
    return value;
  }  // Bucle indefinido hasta que se dé un valor compelto
}

#endif  // __UTILITIES_H__
```

Carpeta src

date.cpp

```
#include "date.hpp"

// Constructor base que usa chorono para inicializar la fecha en hoy
Date::Date() {
  auto now = std::chrono::system_clock::now();
  std::time_t currentTime = std::chrono::system_clock::to_time_t(now);
  std::tm localTime = *std::localtime(&currentTime);

  this->year = localTime.tm_year + 1900;
  this->month = localTime.tm_mon + 1;
  this->day = localTime.tm_mday;
}

// Constructor Copia
Date::Date(const Date& other)
    : year(other.year), month(other.month), day(other.day) {}

// Constructor Paramétrico
Date::Date(const int& y, const int& m, const int& d)
    : year(y), month(m), day(d) {}

// Getter's
int Date::getYear() const {
  return this->year;
}

int Date::getMonth() const {
  return this->month;
}

int Date::getDay() const {
  return this->day;
}

// toString()
std::string Date::toString() const {
  // Utilizamos un arreglo para imprimir la fecha de texto y no solo en
formato
  // dd/mm/aaaa
  std::string months[12] = {"Enero",      "Febrero",
"Marzo",     "Abril",
                            "Mayo",       "Junio",    "Julio",     "Agosto
",
                            "Septiembre", "Octubre", "Noviembre",
"Diciembre"};
```

```
    return std::to_string(day) + " de " + months[month - 1] + " del " +
           std::to_string(year);
}

// Setter's
void Date::setYear(const int& year) {
  auto now = std::chrono::system_clock::now();
  std::time_t timeNow = std::chrono::system_clock::to_time_t(now);
  std::tm dateNow = *std::localtime(&timeNow);

  // Lanzamos una excepción si se intenta poner un año más adelante del
actual o
  // menor a la configuración
  if (year < configure::minimumYear || year > (dateNow.tm_year + 1900)) {
    throw std::out_of_range("El anio debe estar entre el rango " +
                            std::to_string(configure::minimumYear) + " y
" +
                            std::to_string(dateNow.tm_year + 1900));
  }
  this->year = year;
}

void Date::setMonth(const int& month) {
  // Lanzamos excepción si se intenta poner un mes inválido
  if (month < 0 || month > 12)
    throw std::out_of_range(
        "El mes debe estar entre 1 (Enero) y 12 (Diciembre).");
  this->month = month;
}

void Date::setDay(const int& day) {
  // Lanzamos excepción si se intenta poner un día inválido
  if (day < 0 || day > 31)
    throw std::out_of_range("El dia es invalido");
  this->day = day;
}

Date& Date::operator=(const Date& other) {
  this->year = other.year;
  this->month = other.month;
  this->day = other.day;

  return *this;
}
```

inventory.cpp

```
#include "inventory.hpp"

// Constructores
Inventory::Inventory() : numberOfProducts(0) {}
// Constructor Copia
Inventory::Inventory(const Inventory& other)
    : numberOfProducts(other.numberOfProducts) {
  for (int i = 0; i < other.numberOfProducts;
       i++)  // Recorremos el arreglo para copiarlo
    this->products[i] = other.products[i];
}

// constructor paramétrico
Inventory::Inventory(const int& n, const Product* products)
    : numberOfProducts(n) {
  for (int i = 0; i < configure::inventoryCapacity; i++)
    this->products[i] = products[i];
}

// Getter's
int Inventory::getNumberOfProducts() const {
  return this->numberOfProducts;
}

const Product* Inventory::getProducts() const {
  return products;
}

// toString()
std::string Inventory::toString() const {
  std::ostringstream
      oss;  // Usamos iomanip para estilizar el inventario en formato de
tabla

  oss <<
"================================================================="
        "=======================================\n";
  oss <<
"                                                    INVENTARIO "
        "COMPLETO           \n";
  oss <<
"================================================================="
        "=======================================\n";

  // Encabezados
  oss << std::left << std::setw(20) << "CODIGO DE BARRAS" <<
std::setw(10)
```

```
        << "NOMBRE" << std::setw(10) << "PESO" << std::setw(32)
       << "FECHA DE ENTRADA" << std::setw(20) << "PRECIO AL MAYOREO"
       << std::setw(20) << "PRECIO AL MENUDEO" << std::setw(8) <<
"EXISTENCIAS";

  oss << "\n-----------------------------------------------------------
------"
         "-------------------------------------------------\n";

  // Recorrer los Productos e imprimir las características
  for (int i = 0; i < numberOfProducts; i++) {
    oss << std::left << std::setw(20) << products[i].getBarcode()
        << std::setw(10) << products[i].getProductName() << std::setw(10)
        << products[i].getWeight() << std::setw(32)
        << products[i].getEntryDate().toString() << std::setw(20)
        << products[i].getWholesalePrice() << std::setw(20)
        << products[i].getRetailPrice() << std::setw(8)
        << products[i].getCurrentExistence() << "\n";
  }

  oss << "\n\n-----------------------------------------------------------
------"
         "-------------------------------------------------\n";

  oss << std::left << std::setw(20)
      << "TOTAL DE PRODUCTOS: " << numberOfProducts << "\n";

  return oss.str();
}

// Setter's
void Inventory::setNumberOfProducts(const int& numberOfProducts) {
  if (numberOfProducts <
      0)  // Excepción si el número de productos se intenta poner
negativo
    throw std::out_of_range(
        "El numero de productos no puede ser menor o igual a 0");
  this->numberOfProducts = numberOfProducts;
}

void Inventory::setProducts(const Product* products) {
  for (int i = 0; i < configure::inventoryCapacity; i++)
    this->products[i] = products[i];
}

Inventory& Inventory::operator=(const Inventory& other) {
  this->numberOfProducts = other.numberOfProducts;
  for (int i = 0; i < configure::inventoryCapacity; i++)
```

```
    this->products[i] = other.products[i];
  return *this;
}

// Buscador
Product* Inventory::searchProduct(const std::string& barcode) {
  for (int i = 0; i < numberOfProducts; i++)
    if (products[i].getBarcode() == barcode)
      return &products[i];
  return nullptr;
}

// Algorítmicos
void Inventory::addProduct(const Product& newProduct) {
  if (numberOfProducts >=
      configure::inventoryCapacity)  // Excepción si el inventario está
lleno
    throw std::out_of_range("Inventario lleno");

  if (searchProduct(newProduct.getBarcode()) !=
      nullptr)  // Excepción si el producto ya está registrado
    throw std::invalid_argument("El producto ya esta registrado.");

  products[numberOfProducts] =
      newProduct;       // Creamos el producto en el espacio
correspondiente
  numberOfProducts++;  // Aumentamos el número de productos
}

void Inventory::increaseProduct(const std::string& barcode,
                                const int& increase) {
  if (numberOfProducts <=
      0)  // Si no hay productos registrados lanzamos excepción
    throw std::logic_error("No hay productos registrados");

  Product* validation = searchProduct(barcode);
  if (validation ==
      nullptr)  // Si el código de barras no se encuentra lanzamos
excepción
    throw std::invalid_argument("Codigo de barras no encontrado");

  validation->increaseExistence(
      increase);  // el producto aumenta su existencia en la cantidad dad
}

void Inventory::decreaseProduct(const std::string& barcode,
                                const int& decrease) {
  if (numberOfProducts <=
```

```
        0)  // Si no hay productos registrados lanzamos excepción
    throw std::logic_error("No hay productos registrados");

  Product* validation = searchProduct(barcode);
  if (validation ==
      nullptr)  // Si no se encuentre al código de barras lanzamos
excepción
    throw std::invalid_argument("Codigo de barras no encontrado");
  if (decrease >
      validation->getCurrentExistence())  // Si la cantidad que se va a
bajar
                                          // dejaría en negativo las
existencias
                                          // lanzamos excepción
    throw std::invalid_argument(
        "No puedes poner una cantidad que dejaria en negativo el stock");
  validation->decreaseExistence(decrease);
}
```

main.cpp

```
/*Nombre: Mariscal Rodríguez Omar Jesús
Fecha: 24 de Agosto de 2025
Actividad 02: Anidación Estructural
Versión: 1.0*/

#include "menu.hpp"

int main() {
  Menu menu;  // El constructor de menú incializa todo el programa. No
hace
              // falta nada más en el main.

  return 0;
}
```

menu.cpp

```
#include "menu.hpp"

// std's necesarios
using std::cin;
using std::cout;
using std::endl;

void Menu::mainMenu() {  // Menú Principal
  system("CLS");
  std::string menuSelection = "";
  menuSelection += "====================================\n";
  menuSelection += "     MENU PRINCIPAL - INVENTARIO    \n";
  menuSelection += "====================================\n";
  menuSelection += "1. Registrar un Nuevo Producto.\n";
  menuSelection += "2. Aumentar Existencia\n";
  menuSelection += "3. Disminuir Existencia\n";
  menuSelection += "4. Consultar Inventario\n";
  menuSelection += "5. Buscar Producto por Codigo\n";
  menuSelection += "6. Salir\n";
  menuSelection += "------------------------------------\n";
  menuSelection += "Seleccione una opcion: ";

  int selection = Utilities<int>::readNumberRange(
      1, 6, menuSelection);  // Uso de utilities para la lectura de datos
  // Redirección a la pantalla especificada
  switch (selection) {
    case 1:
      newProduct();
      break;
    case 2:
      increaseStock();
      break;
    case 3:
      decreaseStock();
      break;
    case 4:
      checkInventory();
      break;
    case 5:
      searchProduct();
      break;
    case 6:
      exitScreen();
      break;
      // No necesitamos default, al ya depurarase en el dato gracias a
utilities
  }
```

```
}

void Menu::newProduct() {
  system("CLS");

  std::string menuString;
  menuString += "====================================\n";
  menuString += "     REGISTRAR NUEVO PRODUCTO     \n";
  menuString += "====================================\n";

  std::string dataString;
  int dataInt;
  float dataFloat;
  Date entryDate;
  Product newProduct;

  do {  // Asegurarnos que el código de barras es de los caracteres topes
de
        // configure.hpp
    cout << menuString;  // Impresión del Menú
    cout << "Ingrese el codigo de Barras: ";
    std::getline(cin, dataString);

    if (dataString.size() > configure::barcodeSize) {
      system("CLS");
      cout << "El codigo de barras debe ser de " <<
configure::barcodeSize
           << " o menos caracteres.\n";
    }

  } while (dataString.size() > configure::barcodeSize);

  Product* validation = inventory.searchProduct(dataString);

  // Verificar si el producto no está registrado
  if (validation != nullptr) {
    cout << "Producto Ya registado." << endl;
    cout << validation->toString();  // imprimimos el producto para
mostrarlo
    system("PAUSE");
    mainMenu();
  };

  // Pedimos los datos y los vamos agregando a un newProduct con
variables que
  // constantemente reciclamos
  newProduct.setBarcode(dataString);
```

```
   cout << "Ingrese el Nombre del Producto: ";
  std::getline(cin, dataString);
  newProduct.setProductName(dataString);

  dataFloat =
      Utilities<float>::readPositiveNumber("Ingrese el peso del producto:
");
  newProduct.setWeight(dataFloat);

  int select =
      Utilities<int>::readNumberRange(1, 2,
                                      "Para la fecha de entrada, desea
poner "
                                      "la fecha de hoy? (1. Si/ 2. No):
");

  if (select == 2) {  // Si la fecha se ingresa manualmente, sino, ya
tiene la
                      // del día de hoy por el constructor base
    dataInt =
        Utilities<int>::readNumberRange(1, 31, "Ingrese el dia de
ingreso: ");
    entryDate.setDay(dataInt);

    dataInt = Utilities<int>::readNumberRange(
        1, 12, "Ingrese el numero del mes que se ingreso: ");
    entryDate.setMonth(dataInt);

    dataInt = Utilities<int>::readNumberRange(
        configure::minimumYear, entryDate.getYear(),
        "Ingrese el anio en que se ingreso: ");
    entryDate.setYear(dataInt);
    newProduct.setEntryDate(entryDate);
  }

  dataFloat =
      Utilities<float>::readPositiveNumber("Ingrese el precio al mayoreo:
");
  newProduct.setWholesalePrice(dataFloat);

  dataFloat =
      Utilities<float>::readPositiveNumber("Ingrese el precio al menudeo:
");
  newProduct.setRetailPrice(dataFloat);

  dataInt =
      Utilities<int>::readPositiveNumber("Ingrese la cantidad en stock:
");
```

```
    newProduct.setCurrentExistence(dataInt);

    inventory.addProduct(newProduct);  // Agregamos el producto
    cout << "\n\n";
    cout << "-------------------------------------------------------";
    cout << "Producto agregado con exito!\n";
    system("PAUSE");

    mainMenu();
}

void Menu::increaseStock() {
    system("CLS");
    // Regresamos si el inventario está vacío
    if (inventory.getNumberOfProducts() <= 0) {
        cout << "Aun no existen productos registrados." << endl;
        system("PAUSE");
        mainMenu();
    }
    std::string barcode;
    int select;
    do {
        system("CLS");
        std::string menuString;
        menuString += "====================================\n";
        menuString += " INCREMENTAR EL STOCK DE UN PRODUCTO    \n";
        menuString += "====================================\n";

        cout << menuString;

        cout << "Ingrese el codigo de barras del producto: ";
        getline(cin, barcode);
        Product* validation = inventory.searchProduct(barcode);

        // Verificar que el producto exista
        if (validation == nullptr) {
            cout << "Producto no encontrado" << endl;
        }

        else {
            int increase = Utilities<int>::readPositiveNumber(
                validation->toString() +
                "\n Cuantas Unidades desea aumentar el stock?: ");
            inventory.increaseProduct(barcode, increase);  // incremento de
unidades
            cout << "Existencias Agregadas con Exito!" << endl;
        }
        // Preguntar si se desea incrementar otro producto
```

```
        select = Utilities<int>::readNumberRange(
            1, 2, "Desea Ingresar un Nuevo Codigo de Barras? (1. Si/ 2. No):
");

    } while (select != 2);

    mainMenu();  // Volvemos al menú principal
}

void Menu::decreaseStock() {
    system("CLS");
    if (inventory.getNumberOfProducts() <=
        0) {  // Si el inventario está vacío regresamos
        cout << "Aun no se registran productos en el sistema." << endl;
        system("PAUSE");
        mainMenu();
    }

    std::string barcode;
    int select;

    do {
        system("CLS");
        std::string menuString;
        menuString += "====================================\n";
        menuString += "   BAJAR EL STOCK DE UN PRODUCTO    \n";
        menuString += "====================================\n";

        cout << "Ingrese el codigo de barras del producto que desea quitar
stock: ";
        getline(cin, barcode);

        Product* validation = inventory.searchProduct(barcode);
        if (validation == nullptr) {
            cout << "El codigo de barras no esta registrado." << endl;
        } else {
            if (validation->getCurrentExistence() == 0) {
                cout << validation->toString();
                cout << "Las existencias estan en 0, no pueden decrecer.\n";
            } else {
                int decrease = Utilities<int>::readNumberRange(
                    1, validation->getCurrentExistence(),
                    validation->toString() +
                        "\nIngrese la cantidad que desee quitar del stock actual:
");
                inventory.decreaseProduct(barcode, decrease);
                cout << "\nDecremento hecho con exito!\n";
            }
```

```
    }

    select = Utilities<int>::readNumberRange(
        1, 2, "Desea ingresar otro codigo de barras? (1. Si/ 2. No): ");

  } while (select != 2);

  mainMenu();
}

void Menu::checkInventory() {
  system("CLS");
  if (inventory.getNumberOfProducts() <= 0) {  // Si El inventario está
vacío
    cout << "Aun no se Registran Productos." << endl;
    system("PAUSE");
    mainMenu();
  }
  cout << inventory.toString();  // Imprimimos el toString() del
inventario
  system("PAUSE");
  mainMenu();
}

void Menu::searchProduct() {
  system("CLS");
  if (inventory.getNumberOfProducts() <= 0) {
    cout << "No hay productos registrados aun." << endl;
    cout << "Regresando al menu principal." << endl;
    system("PAUSE");
    mainMenu();
  }
  std::string barcode;
  int select;

  do {
    system("CLS");
    std::string menuString;
    menuString += "====================================\n";
    menuString += "           BUSCAR PRODUCTO    \n";
    menuString += "====================================\n";
    cout << menuString;

    cout << "Ingrese el codigo de barras del producto que desea buscar:
";
    getline(cin, barcode);

    Product* validation = inventory.searchProduct(barcode);
```

```
    if (validation == nullptr)
      cout << "El codigo de barras " << barcode
           << " no se encuentra registrado en el sistema.";
    else
      cout << validation->toString();

    select = Utilities<int>::readNumberRange(
        1, 2, "Desea Realizar Otra Busqueda? (1. Si/ 2. No): ");

  } while (select != 2);

  mainMenu();
}

void Menu::exitScreen() {
  system("CLS");
  std::cout << "=======================================\n";
  std::cout << "        Gracias por usar el sistema      \n";
  std::cout << "             INVENTARIO 1.0             \n";
  std::cout << "=======================================\n";
  std::cout << "   Hecho por: Mariscal Rodriguez Omar Jesus \n";
  std::cout << "   Materia: Estructuras de Datos \n";
  std::cout << "   Profesor: Gutierrez Hernandez Alfredo \n";
  std::cout << "   Hasta Pronto! Lindo Dia :D \n";
  std::cout << "=======================================\n\n";
  system("PAUSE");
  return;
}

Menu::Menu() {
  // Constructor llama a la función del menú principal
  mainMenu();
}
```

product.cpp

```cpp
#include "product.hpp"

// Constructor base
Product::Product()
    : barcode(""),
      productName(""),
      weight(0.0),
      entryDate(Date()),
      wholesalePrice(0.0),
      retailPrice(0.0),
      currentExistence(0) {}
// Constructor copia
Product::Product(const Product& other)
    : barcode(other.barcode),
      productName(other.productName),
      weight(other.weight),
      entryDate(other.entryDate),
      wholesalePrice(other.wholesalePrice),
      retailPrice(other.retailPrice),
      currentExistence(other.currentExistence) {}

// Getter's
std::string Product::getBarcode() const {
  return this->barcode;
}

std::string Product::getProductName() const {
  return this->productName;
}

float Product::getWeight() const {
  return this->weight;
}

Date Product::getEntryDate() const {
  return this->entryDate;
}

float Product::getWholesalePrice() const {
  return this->wholesalePrice;
}

float Product::getRetailPrice() const {
  return this->retailPrice;
}

int Product::getCurrentExistence() const {
```

```
    return this->currentExistence;
}

// Impresión de las características de un producto
std::string Product::toString() const {
  std::ostringstream oss;
  oss << "=====================================\n"
      << "       INFORMACION DEL PRODUCTO     \n"
      << "=====================================\n";

  oss << "Codigo de Barras: " << barcode << "\nNombre del Producto "
      << productName << "\nPeso del Producto: " << weight
      << "\nFecha de Ingreso: " << entryDate.toString()
      << "\nPrecio al Mayorista: " << wholesalePrice
      << "\nPrecio al Minorista: " << retailPrice
      << "\nExistencias Actuales: " << currentExistence;
  oss << "\n=====================================\n";

  return oss.str();
}

// Setter's
void Product::setBarcode(const std::string& barcode) {
  // Si el código de barras excede el configure, lanzamos excepción
  if (barcode.size() > configure::barcodeSize)
    throw std::out_of_range("Solo se permiten " +
                            std::to_string(configure::barcodeSize) +
                            " caracteres.");
  this->barcode = barcode;
}

void Product::setProductName(const std::string& productName) {
  // Si el nombre está vacío lanzamos excepción
  if (productName.empty())
    throw std::invalid_argument("El nombre no puede estar vacío");
  this->productName = productName;
}

void Product::setWeight(const float& weight) {
  // Si el peso es negativo o igual a 0 lanzamos excepción
  if (weight <= 0)
    throw std::invalid_argument("El peso no puede ser menor ni igual que
0");
  this->weight = weight;
}

void Product::setEntryDate(const Date& entryDate) {
  this->entryDate = entryDate;  // Date ya tiene sus propias validaciones
```

```
 }

void Product::setWholesalePrice(const float& wholesalePrice) {
  // El precio no puede ser <= 0, lanzamos excepción
  if (wholesalePrice <= 0)
    throw std::invalid_argument("El precio no puede ser menor o igual a
0");
  this->wholesalePrice = wholesalePrice;
}

void Product::setRetailPrice(const float& retailPrice) {
  // El precio no puede ser <=0, lanzamos excepción
  if (retailPrice <= 0)
    throw std::invalid_argument("El precio no puede ser menor o igual a
0");
  this->retailPrice = retailPrice;
}

void Product::setCurrentExistence(const int& currentExistence) {
  // Las exist3encias no pueden ser negativas
  if (currentExistence < 0)
    throw std::invalid_argument(
        "Un producto no puede quedar en cantidad negativa.");
  this->currentExistence = currentExistence;
}

Product& Product::operator=(const Product& other) {
  this->barcode = other.barcode;
  this->productName = other.productName;
  this->weight = other.weight;
  this->entryDate = other.entryDate;
  this->wholesalePrice = other.wholesalePrice;
  this->retailPrice = other.retailPrice;
  this->currentExistence = other.currentExistence;

  return *this;
}

void Product::increaseExistence(const int& increase) {
  this->currentExistence += increase;
}

void Product::decreaseExistence(const int& decrease) {
  if (decrease > currentExistence)  // Lanzamos excepción si las
existencias
                                    // quieren ser negativas.
    throw std::logic_error("No puedes tener valores negativos");
  currentExistence -= decrease;}
```

Ejecución del Programa

Empezamos ejecutando el programa, lo que libera el menú principal:

```
====================================
     MENU PRINCIPAL - INVENTARIO
====================================
1. Registrar un Nuevo Producto.
2. Aumentar Existencia
3. Disminuir Existencia
4. Consultar Inventario
5. Buscar Producto por Codigo
6. Salir
------------------------------------
Seleccione una opcion:
```

Aquí tenemos que ingresar una opción, esta entrada está vigilada con utilities, así que aunque pongamos un número inválido o incluso datos de otro tipo, nos saltará un aviso y pedirá nuevamente:

```
Entrada invalida.
 Por favor, intetelo nuevamente.

Presione una tecla para continuar . . .
```

En el inicio del programa, si intentamos acceder a funciones de consulta o modificación de existencia como la segunda opción, el programa no nos dejará entrar y se mostrará lo siguiente:

```
Aun no existen productos registrados.
Presione una tecla para continuar . . .
```

Por lo que al principio, solo tenemos acceso al registro de productos o a salir del programa, vayamos a la opción 1:

```
====================================
     REGISTRAR NUEVO PRODUCTO
====================================
Ingrese el codigo de Barras:
```

Se nos pide un código de barras, como está limitado por las opciones de configure, si ingresamos un código de más de 13 dígitos, no nos dejará:

```
El codigo de barras debe ser de 13 o menos caracteres.
```

Y nos regresa a la pantalla, si ingresamos un código válido nos seguirá pidiendo datos:

```
==================================
     REGISTRAR NUEVO PRODUCTO
==================================
Ingrese el codigo de Barras: 2530186010584
Ingrese el Nombre del Producto: Zucaritas
Ingrese el peso del producto: 1.3
Para la fecha de entrada, desea poner la fecha de hoy? (1. Si/ 2. No):
```

En la fecha, tenemos la opción de inicializarla al día de hoy o poner una propia, para este primer producto, pongámosla a fecha de hoy:

```
==================================
     REGISTRAR NUEVO PRODUCTO
==================================
Ingrese el codigo de Barras: 2530186010584
Ingrese el Nombre del Producto: Zucaritas
Ingrese el peso del producto: 1.3
Para la fecha de entrada, desea poner la fecha de hoy? (1. Si/ 2. No): 1
Ingrese el precio al mayoreo: 112.5
Ingrese el precio al menudeo: 120.5
Ingrese la cantidad en stock: 10


------------------------------------------------------------Producto agregado con exito!
```

Se nos dice que el producto ha sido agregado con éxito, lo que nos devuelve al menú principal. Agreguemos otro producto con una fecha y stock distinto, nos pedirá el día, y si agregamos algo negativo o sin sentido nos aparecerá:

```
Numero fuera de rango.
 Por favor, intetelo nuevamente.

Ingrese el dia de ingreso:
```

Lo mismo pasará si intentamos indicar un año en el futuro como 2026 o similares,

```
Ingrese el dia de ingreso: 29
Ingrese el numero del mes que se ingreso: 01
Ingrese el anio en que se ingreso: 2025
Ingrese el precio al mayoreo: 32
Ingrese el precio al menudeo: 42.3
Ingrese la cantidad en stock: 5


------------------------------------------------------Producto agregado con exito!
```

Acabando de agregar otro producto, vemos que pasa si intentamos agregar otro producto con el mismo código de barras:

```
====================================
     REGISTRAR NUEVO PRODUCTO
====================================
Ingrese el codigo de Barras: 2530186010584
Producto Ya registado.
========================================
       INFORMACION DEL PRODUCTO
========================================
Codigo de Barras: 2530186010584
Nombre del Producto Zucaritas
Peso del Producto: 1.3
Fecha de Ingreso: 23 de Agosto del 2025
Precio al Mayorista: 112.5
Precio al Minorista: 120.5
Existencias Actuales: 10
========================================
Presione una tecla para continuar . . .
```

Nos dirá que el producto ya está registrado y nos muestra a qué elemento está asociado ese código de barras; el aumento y decremento lo manejamos desde el menú principal.

Ahora, vayamos a la opción de aumentar existencias del menú principal:

```
====================================
 INCREMENTAR EL STOCK DE UN PRODUCTO
====================================
Ingrese el codigo de barras del producto:
```

Nos pide un código de barras y al ingresar uno existente (uno inexistente simplemente nos avisará)

```
====================================
 INCREMENTAR EL STOCK DE UN PRODUCTO
====================================
Ingrese el codigo de barras del producto: 2530186010584
========================================
       INFORMACION DEL PRODUCTO
========================================
Codigo de Barras: 2530186010584
Nombre del Producto Zucaritas
Peso del Producto: 1.3
Fecha de Ingreso: 23 de Agosto del 2025
Precio al Mayorista: 112.5
Precio al Minorista: 120.5
Existencias Actuales: 10
========================================

 Cuantas Unidades desea aumentar el stock?:
```

Nos muestra la información del producto y las existencias actuales, cantidades negativas nos mostrará un pequeño mensaje:

```
Numero fuera de rango.
 Por favor, intetelo nuevamente.
```

Aumentemos las existencias en 10, por lo que debería ascender hasta 20

```
=======================================
       INFORMACION DEL PRODUCTO
=======================================
Codigo de Barras: 2530186010584
Nombre del Producto Zucaritas
Peso del Producto: 1.3
Fecha de Ingreso: 23 de Agosto del 2025
Precio al Mayorista: 112.5
Precio al Minorista: 120.5
Existencias Actuales: 10
=======================================

 Cuantas Unidades desea aumentar el stock?: 10
Existencias Agregadas con Exito!
Desea Ingresar un Nuevo Codigo de Barras? (1. Si/ 2. No):
```

Nos avisa que se agregaron con éxito, y si queremos ingresar otro código de barras, si le damos que sí, se repetirá el menú.

Ahora vayamos a la opción 3, para el quitar existencias:

```
Ingrese el codigo de barras del producto que desea quitar stock: 2656389458124
=======================================
       INFORMACION DEL PRODUCTO
=======================================
Codigo de Barras: 2656389458124
Nombre del Producto Frijol
Peso del Producto: 1.3
Fecha de Ingreso: 29 de Enero del 2025
Precio al Mayorista: 32
Precio al Minorista: 42.3
Existencias Actuales: 5
=======================================

Ingrese la cantidad que desee quitar del stock actual:
```

De la misma manera, nos muestra la información y las existencias actuales del producto, y si intentamos quitar algo que dejaría en negativo como 6:

```
Numero fuera de rango.
 Por favor, intetelo nuevamente.
```

Nos avisa que el valor está fuera de rango. De este segundo producto quitemos 3, para que queden 2 en existencias

```
Decremento hecho con exito!
Desea ingresar otro codigo de barras? (1. Si/ 2. No):
```

Ahora, pasemos con la opción 4 del inventario, para visualizarla mejor, agreguemos otros dos productos al inventario con la opción 1.

```
====================================
     REGISTRAR NUEVO PRODUCTO
====================================
Ingrese el codigo de Barras: 2634598555641
Ingrese el Nombre del Producto: Cafe
Ingrese el peso del producto: 200
Para la fecha de entrada, desea poner la fecha de hoy? (1. Si/ 2. No): 2
Ingrese el dia de ingreso: 03
Ingrese el numero del mes que se ingreso: 6
Ingrese el anio en que se ingreso: 2021
Ingrese el precio al mayoreo: 67.5
Ingrese el precio al menudeo: 85.3
Ingrese la cantidad en stock: 10


------------------------------------------------------------Producto agregado con exito!
```

```
====================================
     REGISTRAR NUEVO PRODUCTO
====================================
Ingrese el codigo de Barras: 1326544856235
Ingrese el Nombre del Producto: Arroz
Ingrese el peso del producto: 1.0
Para la fecha de entrada, desea poner la fecha de hoy? (1. Si/ 2. No): 1
Ingrese el precio al mayoreo: 9
Ingrese el precio al menudeo: 15
Ingrese la cantidad en stock: 32


------------------------------------------------------------Producto agregado con exito!
```

Ahora, dándole a la opción 4 nos imprime todo el inventario de esta manera:

```
================================================================================================================================
                                                INVENTARIO COMPLETO
================================================================================================================================
CODIGO DE BARRAS    NOMBRE    PESO      FECHA DE ENTRADA              PRECIO AL MAYOREO   PRECIO AL MENUDEO   EXISTENCIAS
--------------------------------------------------------------------------------------------------------------------------------
2530186010584       Zucaritas 1.3       23 de Agosto del 2025         112.5               120.5               20
2656389458124       Frijol    1.3       29 de Enero del 2025          32                  42.3                2
2634598555641       Cafe      200       3 de Junio del 2021           67.5                85.3                10
1326544856235       Arroz     1         23 de Agosto del 2025         9                   15                  32


--------------------------------------------------------------------------------------------------------------------------------
TOTAL DE PRODUCTOS: 4
Presione una tecla para continuar . . .
```

Nos da una tabla con todas las características de todos los productos registrados, vemos que se ven reflejados los cambios en los primeros productos justo como planeábamos, las existencias subieron a 20 y bajaron a 2 respectivamente, confirmando que las funciones de aumentar y disminuir las existencias funcionan.

Pasando con la opción 5, simplemente es un buscador, ingresamos el código y nos imprime la información de un producto.

```
====================================
          BUSCAR PRODUCTO
====================================
Ingrese el codigo de barras del producto que desea buscar: 2634598555641
======================================
       INFORMACION DEL PRODUCTO
======================================
Codigo de Barras: 2634598555641
Nombre del Producto Cafe
Peso del Producto: 200
Fecha de Ingreso: 3 de Junio del 2021
Precio al Mayorista: 67.5
Precio al Minorista: 85.3
Existencias Actuales: 10
======================================
Desea Realizar Otra Busqueda? (1. Si/ 2. No):
```

Finalmente, al darle a la opción 6 del menú principal, nos muestra el mensaje de salida:

```
========================================
        Gracias por usar el sistema
             INVENTARIO 1.0
========================================
   Hecho por: Mariscal Rodriguez Omar Jesus
   Materia: Estructuras de Datos
   Profesor: Gutierrez Hernandez Alfredo
   Hasta Pronto! Lindo Dia :D
========================================

Presione una tecla para continuar . . .
```

Y así el programa finaliza.

Conclusiones

Este trabajo fue un repaso bastante interesante, muchas de las materias de anteriores semestres tenían como proyecto final algo similar a esto, un inventario, y haciéndolo en este punto veo un gran avance en conceptos, en mi entendimiento del lenguaje y la eficiencia en la que hago ciertas cosas; fue una recapitulación express de la carrera hasta ahora, aunque ya he trabajado con memoria dinámica con herramientas como realloc en c o la biblioteca vector en C++, hacerlo de manera estática manejando los índices donde deberían ir controlados por una clase es llamativo.

Apliqué bastantes conceptos que aprendí de las últimas clases y se puede revisar en cómo cambió mi trabajo, como en las inclusiones, antes las ponía en las definiciones, pero es más limpio y entendible ponerlas en los encabezados para luego solo tener que incluir este en el .cpp. Otra cosa implementé fue tener validaciones en los setter's, por medio de excepciones, usualmente manejaba las depuraciones fuera de los mismos objetos, pero para mantener un correcto encapsulamiento es cierto que es mejor tener validaciones ahí, aunque datos se depuren fuera de la propia clase. Además de esto, en clase se mencionó que un objeto debe de tener sus funcionalidades básicas como los constructores base, copia y paramétricos, los toString() y las interfaces, a pesar de que no los usemos, era una cosa que no implementaba si no los necesitaba, no los programaba. En este caso, seguí más a detalle estas indicaciones como una buena práctica.

Con respecto a la anidación, tener composiciones de objetos es algo común que ya hacía, muy útil y que requiere un nivel un poco más alto de abstracción para ver un dato como la composición de varios datos, y más aún, un arreglo de objetos compuestos por otros objetos, pero desde una perspectiva correcta es muy manejable.

En síntesis, fue una actividad enriquecedora que, además de funcionar como un repaso, pude poner en práctica conceptos de POO que vamos rescatando clase a clase, que además de hacer un código más elegante, mejoran y anticipan más situaciones sobre cuales los objetos deben estar fundamentados.