Informe de Prácticas

**LENGUAJES DE PROGRAMACIÓN II**

**UCSM Esc. Prof. de Ingeniería de Sistemas**

**Marzo - 2025**



**Práctica N° 12: Standard Template Library en C++**

Elaborado por:

HUANQUI LUQUE PIEROL YARENAPELLIDOS

SANCHEZ YDME RODRIGO FABRIZIO



**GRUPO N° 0X**

**PRÁCTICAS DE SISTEMAS INTELIGENTES**

Presentado por:

|  |  |  |
| --- | --- | --- |
| 2024002019 | HUANQUI LUQUE PIEROL YAREN | 100% |
| 2024002172 | SANCHEZ YDME RODRIGO FABRIZIO | 100% |

**RECONOCIMIENTOS**

Se reconoce la importancia de la STL como una herramienta poderosa que simplifica y optimiza el proceso de desarrollo en C++. La comprensión y correcta aplicación de sus componentes contribuyen a la creación de programas eficientes, reutilizables y fáciles de mantener. La formación en esta biblioteca es esencial para resolver problemas de programación complejos y para potenciar las habilidades de los futuros ingenieros de sistemas.

**PALABRAS CLAVES**

STL, Contenedores, Algoritmos, Iteradores, Programación en C++, Eficiencia, Reutilización, Datos, Estructuras de Datos, Programación Orientada a Datos.

**ÍNDICE**

[1. RESUMEN 1](#_Toc201871783)

[2. INTRODUCCIÓN 1](#_Toc201871784)

[3. INFOGRAFÍA 1](#_Toc201871785)

[4. ACTIVIDADES 2](#_Toc201871786)

[5. EJERCICIOS 5](#_Toc201871787)

[7. CUESTIONARIO 10](#_Toc201871788)

[8. BIBLIOGRAFÍA **¡Error! Marcador no definido.**](#_Toc201871789)

**ÍNDICE DE TABLAS Y FIGURAS**

[Codigo N° 1 2](#_Toc201871940)

[Codigo N° 2 3](#_Toc201871941)

[Codigo N° 3 4](#_Toc201871942)

[Codigo N° 4 5](#_Toc201871943)

[Codigo N° 5 6](#_Toc201871944)

[Codigo N° 6 7](#_Toc201871945)

[Codigo N° 7 8](#_Toc201871946)

[Codigo N° 8 9](#_Toc201871947)

# RESUMEN

La práctica de la Sesión 12 de Programación II se centra en la utilización de la Biblioteca Estándar de Plantillas (STL) en C++, abordando sus componentes principales: contenedores, algoritmos e iteradores. A través de ejemplos y ejercicios prácticos, los estudiantes aprenden a manipular datos de forma eficiente y a resolver problemas comunes de programación utilizando las estructuras y funciones predefinidas de la STL. La sesión busca fortalecer las habilidades en programación eficiente, facilitando el desarrollo de soluciones reutilizables y optimizadas en C++

# INTRODUCCIÓN

La STL en C++ es una biblioteca fundamental que proporciona agrupar estructuras de datos y algoritmos, permitiendo a los programadores gestionar y manipular datos de manera efectiva. En la presente sesión, se explorarán los diferentes tipos de contenedores, algoritmos y el uso de iteradores, facilitando la comprensión y aplicación práctica de estas herramientas. La inversión en el aprendizaje de la STL permite mejorar la productividad, garantizar código más eficiente y promover buenas prácticas en programación.

# INFOGRAFÍA



Infografia

# ACTIVIDADES

**Experiencia de Práctica N° 01: Manipulación de Contenedores** Objetivo: Utilizar los contenedores de la STL para almacenar y manipular datos. Descripción: Crear un programa que permita ingresar una lista de números enteros y almacenarlos en un vector. Luego, realizar operaciones como calcular la suma, encontrar el máximo y mínimo, ordenar de forma ascendente y eliminar los números pares. Mostrar los resultados y el contenido final del contenedor.



Codigo N°

**Experiencia de Práctica N° 02: Uso de Algoritmos** Objetivo: Aplicar los algoritmos de la STL para resolver problemas específicos. Descripción: Implementar un programa que lea palabras ingresadas por el usuario y las almacene en una lista. Luego, utilizar los algoritmos sort para ordenar, count para contar apariciones de una palabra, y transform para convertir las palabras a mayúsculas. Después, mostrar la lista modificada.



Codigo N°

**Experiencia de Práctica N° 03: Manipulación de Iteradores** Objetivo: Utilizar los iteradores para acceder y modificar elementos de los contenedores. Descripción: Crear un programa que permita ingresar una cadena de caracteres y almacenarla en un contenedor deque. Los estudiantes deben aprender a recorrer, modificar y eliminar elementos utilizando diferentes tipos de iteradores.



Codigo N°

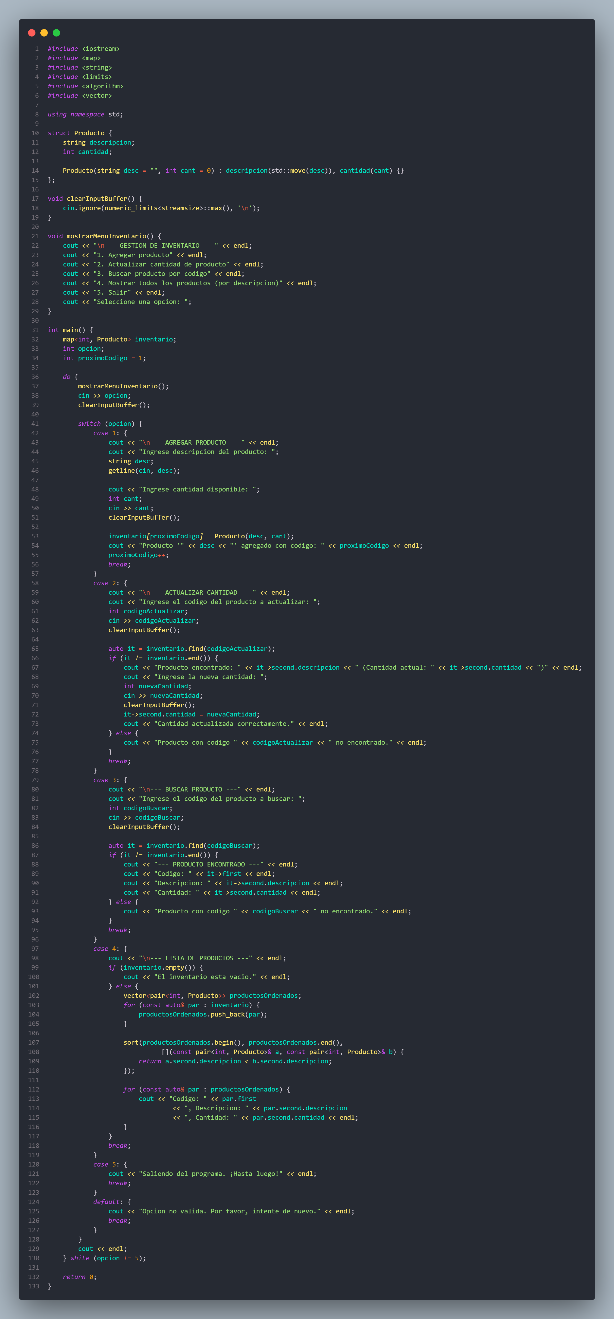
# EJERCICIOS

**Problema: Gestión de Inventario**

Una empresa necesita gestionar su inventario de productos. Cada producto tiene un código único, una descripción y una cantidad disponible en stock.  
El objetivo es desarrollar un programa que permita realizar las siguientes operaciones:

* a) Agregar un nuevo producto al inventario.
* b) Actualizar la cantidad disponible de un producto existente.
* c) Buscar un producto por su código.
* d) Mostrar la lista de productos ordenada alfabéticamente por su descripción.

Utiliza la STL para implementar la solución y proporciona una interfaz amigable para que los usuarios puedan interactuar con el programa.



Codigo N°

**2. Problema: Organización de Eventos**

Un organizador de eventos necesita un sistema para gestionar la lista de asistentes a diferentes eventos. Cada evento tiene un nombre, una fecha y una lista de asistentes.  
El objetivo es desarrollar un programa que permita realizar las siguientes operaciones:

* a) Agregar un nuevo evento a la lista.
* b) Agregar asistentes a un evento específico.
* c) Mostrar la lista de asistentes de un evento en orden alfabético.
* d) Buscar eventos por fecha.

Utiliza la STL para implementar la solución y proporciona una interfaz intuitiva para que el organizador pueda gestionar los eventos y los asistentes de manera eficiente.



Codigo N°



Codigo N°

1. **3. Problema: Registro de Contactos**

Una agenda de contactos requiere un programa que permita almacenar y gestionar los datos de contactos de una persona. Cada contacto tiene un nombre, número de teléfono y dirección de correo electrónico.  
El objetivo es desarrollar un programa que permita realizar las siguientes operaciones:

* a) Agregar un nuevo contacto a la agenda.
* b) Buscar un contacto por su nombre.
* c) Actualizar la información de un contacto existente.
* d) Mostrar la lista de contactos ordenada alfabéticamente.

Utiliza la STL para implementar la solución y proporciona una interfaz sencilla para que los usuarios puedan administrar eficazmente su lista de contactos.



Codigo N°



Codigo N°

# CUESTIONARIO

**1. ¿Qué significa STL en C++?**  
STL significa *Standard Template Library*, una biblioteca de plantillas que proporciona estructuras de datos y algoritmos genéricos.

**2. ¿Cuáles son los componentes principales de la STL?**  
Los principales componentes son: contenedores, algoritmos e iteradores.

**3. ¿Cuál es la utilidad de los contenedores en la STL?**  
Permiten almacenar y gestionar colecciones de datos de manera eficiente y flexible.

**4. ¿Qué tipo de contenedor de la STL se utiliza cuando se necesita una colección ordenada de elementos únicos?**  
El contenedor set.

**5. Menciona tres ejemplos de contenedores asociativos de la STL.**  
set, map y unordered\_map.

**6. ¿Cuál es la diferencia entre los contenedores vector y list en la STL?**  
vector almacena elementos en un arreglo dinámico contiguo y permite acceso rápido por índice, mientras que list es una lista doblemente enlazada, más eficiente para inserciones/borrados en cualquier posición.

**7. ¿Qué es un iterador en la STL?**  
Es un objeto que permite recorrer los elementos de un contenedor de forma similar a un puntero.

**8. ¿Cuál es la función de los algoritmos en la STL?**  
Realizar operaciones sobre rangos de datos como búsqueda, ordenamiento, conteo, copia, etc.

**9. Menciona tres ejemplos de algoritmos disponibles en la STL.**  
sort, find, count.

**10. ¿Cuál es la ventaja de utilizar los algoritmos de la STL en lugar de implementarlos manualmente?**  
Mayor eficiencia, menor posibilidad de errores y ahorro de tiempo en la programación.

**11. ¿Qué es un iterador de inserción en la STL y cómo se utiliza?**  
Es un iterador especial que permite insertar elementos en un contenedor durante operaciones como copy. Ej.: back\_inserter(vec).

**12. ¿Cuál es la diferencia entre un iterador constante y un iterador normal en la STL?**  
El iterador constante (const\_iterator) no permite modificar el valor apuntado; el normal (iterator) sí.

**13. ¿Qué contenedor de la STL se utiliza para almacenar elementos en pares clave-valor?**  
map o unordered\_map.

**14. Menciona tres ejemplos de algoritmos numéricos disponibles en la STL.**  
accumulate, adjacent\_difference, partial\_sum.

**15. ¿Cuál es la utilidad de los iteradores de flujo en la STL?**  
Permiten leer y escribir datos directamente en flujos de entrada/salida como archivos o consola.

**16. ¿Cuál es la función del adaptador de iterador reverse\_iterator en la STL?**  
Permite recorrer un contenedor en sentido inverso.

**17. ¿Qué contenedor de la STL se utiliza para almacenar elementos en orden de inserción, búsquedas y duplicados de forma rápida?**  
multiset o unordered\_multiset.

**18. ¿Cuál es la diferencia entre el contenedor set y el contenedor map en la STL?**  
set almacena solo claves únicas, mientras que map almacena pares clave-valor con claves únicas.

**19. ¿Qué es el contenedor queue en la STL y cuándo se utiliza?**  
Es una estructura FIFO (primero en entrar, primero en salir), útil para gestionar tareas en orden.

**20. ¿Cuál es la función del contenedor queue en la STL y cómo se implementa?**  
Sirve para manejar datos en orden FIFO y se implementa con std::queue<T>.

**21. ¿Cuál es la función del contenedor list en la STL y cómo se utiliza?**  
Permite inserciones/borrados eficientes en cualquier parte de la lista. Se usa como std::list<T>.

**22. ¿Qué es la función lambda en la STL y cuál es su utilidad?**  
Es una función anónima usada comúnmente para pasar funciones a algoritmos de STL.

**23. ¿Cuál es la diferencia entre vector y stack en la STL y cómo se implementa?**  
vector es un arreglo dinámico, stack es LIFO (último en entrar, primero en salir). stack se implementa como std::stack<T>.

**24. ¿Qué es un iterador con valor y cómo se utiliza en la STL?**  
No es un término estándar en STL. Probablemente se refiere a un iterador que permite acceso y modificación del valor.

**25. ¿Cuál es el algoritmo sort de la STL y cómo se utiliza?**  
sort ordena un rango de elementos. Ej.: std::sort(vec.begin(), vec.end());