Algoritmos y Estructuras de Datos I

Trabajo Práctico N°9	Unidad 9
Modalidad: Semi -Presencial	**Estratégica Didáctica:** Trabajo individual
Metodología de Desarrollo: acordar	**Metodología de Corrección:** acordar docente
Carácter de Trabajo: Obligatorio – Con Nota	**Fecha Entrega:** A confirmar por el Docente.

MARCO TEÓRICO

1. ¿Qué entiende Programas multiarchivo y compilación separada?

Los programas multiarchivo se refieren a aquellos en los que el código fuente está dividido en múltiples archivos o módulos. Cada módulo contiene sus propias declaraciones y subprogramas, lo que facilita una organización modular y clara del proyecto. En la compilación separada, cada módulo puede ser compilado de forma independiente, generando archivos de código objeto que se enlazan posteriormente para formar el ejecutable final, lo que optimiza el tiempo de compilación al recompilar solo los archivos modificados.

2. Describir el concepto de Interfaz frente a implementación

La interfaz en programación se refiere a lo que un módulo o clase ofrece al usuario, sin revelar cómo se implementan esos servicios. La implementación, en cambio, es cómo los subprogramas o métodos llevan a cabo el trabajo especificado. En la programación modular, la interfaz suele estar en un archivo de cabecera (.h) mientras que la implementación está en un archivo de código (.cpp), lo que permite que otros módulos usen las funciones sin conocer los detalles de su funcionamiento.

3. ¿Qué utilidad tienen los módulos de biblioteca, que archivos los componen y representan cada uno?

Los módulos de biblioteca permiten organizar el código en componentes reutilizables. Están compuestos por archivos de cabecera (.h), que contienen declaraciones y prototipos, y archivos de implementación (.cpp), que incluyen el código que realiza el trabajo. El programa principal incluye el archivo de cabecera para utilizar las funciones de la biblioteca sin preocuparse por los detalles de la implementación.

4. ¿Como se realiza la Compilación de programas multiarchivo?

La compilación de programas multiarchivo se realiza compilando cada archivo de código fuente (.cpp) por separado en archivos objeto (.obj). Luego, un ligador combina estos archivos objeto para formar el ejecutable. En G++, se listan los archivos .cpp en el comando de compilación. Si un archivo se modifica, solo ese archivo debe recompilarse, acelerando el proceso de desarrollo.

5. ¿Que función tiene El preprocesador?, dar un ejemplo

El preprocesador tiene la función de realizar operaciones sobre el código antes de la compilación una de sus principales tareas es incluir el contenido de archivos de cabecera en los archivos de código fuente mediante directivas como #include esto permite que el código esté listo para ser compilado al incorporar las declaraciones necesarias para que el compilador verifique el uso correcto de funciones clases o constantes

En el código proporcionado el archivo main.cpp utiliza la directiva #include "lista.h" para incluir las declaraciones de funciones y tipos de datos que se encuentran en el archivo de cabecera lista.h esto permite que las funciones y estructuras definidas en ese archivo puedan ser utilizadas en el archivo principal sin que sea necesario reescribir sus declaraciones

6. ¿Qué entiende x programación modular y en que beneficia?

La programación modular se entiende como una técnica que divide un programa en varios módulos independientes que contienen sus propias declaraciones y subprogramas cada módulo es una unidad funcional que maneja una tarea o funcionalidad específica dentro del programa un módulo puede incluir estructuras de datos o funciones utilitarias

Esta técnica permite que el código fuente se divida en varios archivos cada archivo o módulo puede tener funciones o definiciones relacionadas como cálculos manejo de archivos o listas

Los programas modulares permiten la compilación independiente de cada módulo esto significa que cuando se modifica un archivo solo es necesario recompilar ese archivo en particular sin afectar a los demás los módulos compilados se enlazan después para formar el programa final

La programación modular facilita la gestión de programas grandes al permitir la división en partes más manejables también mejora la reutilización de código y simplifica el trabajo en equipo ya que diferentes miembros pueden trabajar en distintos módulos además la compilación separada acelera el proceso de desarrollo ya que no es necesario recompilar todo el programa cuando se hacen cambios

7. Describir el Problema de las inclusiones múltiples

El problema de las inclusiones múltiples ocurre cuando un archivo de cabecera es incluido más de una vez en un programa, lo que puede causar duplicación de definiciones. Para evitar esto, se usan guardas de inclusión (#ifndef, #define, #endif), asegurando que un archivo de cabecera solo se procese una vez.

8. ¿Que entiende x Compilación condicional?

La compilación condicional en C++ se refiere a la posibilidad de incluir o excluir ciertas secciones de código durante el proceso de compilación, dependiendo de si se cumplen ciertas condiciones predefinidas. Este mecanismo es controlado mediante directivas del preprocesador.

9. ¿Cómo nos protegemos frente a inclusiones múltiples?

Para protegernos de las inclusiones múltiples, se usan directivas de preprocesador como #ifndef, #define y #endif, que aseguran que un archivo de cabecera solo sea incluido una vez, evitando duplicaciones y errores por definiciones repetidas.

10. ¿Que entiende por Espacios de nombres?

Los espacios de nombres organizan variables y funciones en diferentes contextos para evitar conflictos. Un ejemplo es tener un espacio de nombres para una lista ordenada y otro para una lista desordenada, ambos con funciones similares como insertar, pero con implementaciones diferentes. Esto permite usar el operador :: para acceder a las funciones según el contexto necesario.

11. Dar ejemplos de Espacios de Nombres y su Utilidad.

Dar ejemplos de Espacios de Nombres y su Utilidad
Definición de Espacios de Nombres:
Los espacios de nombres en C++ permiten organizar variables y funciones en diferentes contextos para evitar conflictos de nombres dentro de un programa. Son útiles cuando existen varias funciones o variables con el mismo nombre, pero que realizan diferentes tareas en contextos distintos.

Utilidad de los Espacios de Nombres:
Los espacios de nombres son fundamentales en proyectos de gran escala o cuando se usan múltiples bibliotecas. Su principal beneficio es evitar colisiones de nombres y mantener el código más organizado y escalable. Además, con la palabra clave using, se puede simplificar el acceso a los elementos de un espacio de nombres, sin necesidad de usar constantemente el operador ::.

Ejemplo de Implementación:
Un caso típico es tener dos espacios de nombres: uno para una lista ordenada (ordenado) y otro para una lista desordenada (desordenado). Ambas listas pueden tener funciones como insertar y mostrar, pero con implementaciones diferentes según el tipo de lista. Dependiendo del contexto, el programador puede acceder a las funciones utilizando ordenado::insertar o desordenado::insertar.

Uso del Operador de Resolución de Ámbito:
El operador :: permite acceder a las funciones o variables dentro de un espacio de nombres específico. Por ejemplo, si una función está definida dentro del espacio de nombres std, se accede a ella como std::abs. Esto facilita la organización del código y es especialmente útil cuando se trabajan con múltiples bibliotecas o funciones con nombres similares.

MARCO PRÁCTICO

Tomando como Marco el Programa de la Unidad 7 (copiado al final) modificando Modularizándolo mediante el uso de Bibliotecas.
Tener en Cuenta:

- Modularizar el Programa.
- Proteger contra Inclusiones Múltiples.
- Aplicar Espacios de Nombres.
- Aplica apropiadamente los conceptos de abstracción, encapsulación y ocultamiento de información.
- Realiza una apropiada distribución de responsabilidades entre las entidades del espacio de la solución.
- Desarrolla para reusar.
- Reusa apropiadamente las entidades desarrolladas en el espacio curricular.
- Demuestra un uso apropiado de la sintaxis y semántica del lenguaje de programación C++.

COPIA UNIDAD 7
Desarrollar un Programa que:

a. La Directora nos Encarga llevar un registro de los Alumnos por carrera y año . Para ello deberemos armar un Programa que guarde los datos de los Alumnos en Alumnos.txt. Adicionalmente deberá realizar las siguientes Tareas:

1) Mostrar Listado de los Alumnos x pantalla
2) Mostrar un Alumno Determinado (buscar y mostrar x plantalla)
3) Insertar un Alumno
4) Eliminar un Alumno
5) Buscar un alumno
6) Que permita Ordenar de Forma Ascendente y Descendente los Alumnos y
7) Mostrarlos x pantalla
8) Opcional - Que permita elegir el campo de Ordenamiento.

b. **Algunas Consideraciones:**

1) Usar Archivo para persistir y recuperar
2) Usar Estructuras, contador y arrays (max 100 alumnos)
3) Usar sobrecarga de Operadores
bool operator>(tRegistro opIzq, tRegistro opDer);
bool operator<(tRegistro opIzq, tRegistro opDer);
4) Modularizar en funciones la implementación
5) Mantener el Main lo mas pequeño posible.

Lic. Oemig José Luis.