**Instituto Superior de Formación Técnica Nº 151 logo151-trans.gif  
Carrera: Analista de Sistemas  
1 Año. Algoritmos y Estructuras de Datos I**

|  |  |
| --- | --- |
| **Trabajo Práctico Nº 8** | **Unidad 8** |
| **Modalidad:** Semi-Presencial | **Estratégica Didáctica:** Trabajo Grupal. |
| **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?
2. Describir el concepto de Interfaz frente a implementación
3. ¿Qué utilidad tienen los módulos de biblioteca, qué archivos los componen y representan cada uno?
4. ¿Cómo se realiza la Compilación de programas multiarchivo?
5. ¿Qué función tiene el preprocesador? Dar un ejemplo.
6. ¿Qué entiende por programación modular y en qué beneficia?
7. Describir el Problema de las inclusiones múltiples
8. ¿Qué entiende por Compilación condicional?
9. ¿Cómo nos protegemos frente a inclusiones múltiples?
10. ¿Qué entiende por Espacios de nombres?
11. Dar ejemplos de Espacios de Nombres y su Utilidad.

**Marco Practico:**

**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.  
. Aplicar apropiadamente los conceptos de abstracción, encapsulación y ocultamiento de información.  
. Realizar 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:

1. 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 por pantalla)
   3. Insertar un Alumno
   4. Eliminar un Alumno
   5. Buscar un alumno
   6. Que permita Ordenar de Forma Ascendente y Descendente los Alumnos y Mostrarlos por pantalla
   7. Opcional - Que permita elegir el campo de Ordenamiento.

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 más pequeño posible.

Lic. Oemig José Luis.

**Marco Teórico:**

1. ¿Qué entiende Programas multiarchivo y compilación separada?

Un programa multiarchivo es aquel que está compuesto por diversos archivos. Entre estos se distinguen aquellos referidos a interfaces y a implementación. La idea es compilar separadamente estos archivos. De ese modo no sólo se consigue una mayor modularidad y facilidad en la reutilización del código, sino que permite modificar la implementación sin tocar la interfaz.

1. Describir el concepto de Interfaz frente a implementación.

En busca de cumplir con algunas de las premisas en cuanto a encapsulación y ocultamiento de información es aconsejable separar la información de ***lo que se hace*** (interfaz) separado de ***cómo se hace*** (implementación). Justamente, en la interfaz se guarda toda la información referida a constantes, tipos y prototipos, los cuales indican cómo se usan las distintas estructuras y cómo se comunican. Por su parte, es en la implementación donde queda plasmada la *forma* en que se hacen las cosas.

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

Los módulos de biblioteca son códigos fuentes separados, esto es, unidades funcionales completas que pueden funcionar de manera aislada sin llamar a otro módulo. Cada uno consta de una declaración y una implementación.

Estos módulos pueden estar compuestos por varios archivos, cada uno de los cuales tiene una funcionalidad asociada, por ejemplo: un archivo para manejo de listas, otro archivo donde va el programa principal, otro para realizar cálculos, otro para manejo de archivos, etc.

1. ¿Cómo se realiza la Compilación de programas multiarchivo?

Se precompila primero cada archivo de los citados en el punto anterior como archivo .obj y luego se enlazan todos estos archivos juntamente con el principal y algunas otras bibliotecas standard como iostream, fstream, etc, conformando un único ejecutable.

1. ¿Qué función tiene el preprocesador? Dar un ejemplo.

El preprocesador de C (cpp) es el primer programa invocado por el compilador. Este interpreta directivas y genera un único archivo temporal con todo el código del módulo o programa. Procesa directivas como #include, #define e #if

1. ¿Qué entiende por programación modular y en qué beneficia?

La programación modular alude a que el programa está dividido en distintos módulos o unidades funcionales. El beneficio de este estilo de programación es el permitir distribuir las funcionalidades a diferentes personas o grupos de trabajo (distribuir tareas).

1. Describir el Problema de las inclusiones múltiples

Al trabajar en programación multiarchivos es frecuente que más de un módulo llame a una misma biblioteca. Al querer linkear todos estos módulos en el archivo ejecutable, el compilador encuentra que está llamando a una misma biblioteca más de una vez y se produce un error.

1. ¿Qué entiende por Compilación condicional?

Se entiende por compilación condicional a una serie de directivas que se utilizan en l

para evitar los errores descritos en el punto anterior. Básicamente, lo que hacen estas instrucciones es decirle al compilador que, si una determinada biblioteca ya fue incluida, que no la vuelva a incluir. O, que si una biblioteca necesaria todavía no fue incluida, entonces le dice que la incluya.

1. ¿Cómo nos protegemos frente a inclusiones múltiples?

Para protegernos sobre inclusiones múltiples, se utiliza la Compilación Condicional, mediante directivas para el preprocesador como #ifndef #define, etc.

Su uso se describe a continuación:

Directivas *#ifdef*, *#ifndef*, *#else* y *#endif*

Se usan en conjunción con la directiva #define

*#define X #define X*

*#ifdef X #ifndef X*

... // Código if ... // Código if

*[#else [#else*

... // Código else ... // Código else

*] ]*

*#endif #endif*

La directiva *#define* define un símbolo (identificador)

Izquierda: se compilará el “Código if” y no el “Código else”

Derecha: al revés, o nada si no hay else

Las cláusulas *else* son opcionales