Universidad Autónoma deBaja California

Facultad de Ingeniería, Arquitectura yDiseño

Ingeniero en Computación

Asignatura:

Programación Estructurada

Actividad 12:

Archivos de Texto

Brayan Arturo Rocha Meneses

Matricula:

371049

Ensenada Baja California 9 de Noviembre del 2023

Introducción:

Los archivos de texto desempeñan un papel crucial en la manipulación de datos en muchos programas informáticos, y C++ ofrece un conjunto robusto de herramientas para trabajar con ellos. La capacidad de leer y escribir archivos de texto no solo amplía las funcionalidades de un programa, sino que también facilita la persistencia de datos, permitiendo que la información se almacene y recupere de manera eficiente.

Competencia:

La competencia en el manejo de archivos de texto en C++ implica la capacidad de realizar operaciones eficientes de lectura y escritura, así como la manipulación de datos almacenados en estos archivos. Esto incluye el manejo adecuado de errores, la gestión de punteros y la comprensión de los modos de apertura de archivos

Fundamentos:

Apertura de Archivos:

Los archivos de texto se abren en C++ mediante el uso de la clase fstream, que proporciona funcionalidades tanto de entrada como de salida. Los modos de apertura, como ios::in para lectura y ios::out para escritura, son esenciales para definir la naturaleza de la operación.

Lectura y Escritura:

La lectura y escritura de datos en archivos de texto se realiza utilizando operadores de flujo (<< para escribir y >> para leer). Se pueden utilizar ciclos y funciones específicas para realizar operaciones más avanzadas, como la lectura de líneas completas o la escritura formateada.

Manipulación de Punteros:

El manejo adecuado de punteros es crucial al trabajar con archivos. Se deben cerrar los archivos después de su uso y gestionar los punteros de manera responsable para evitar pérdidas de memoria.

Validación y Manejo de Errores:

La competencia en el manejo de archivos incluye la capacidad de validar operaciones, verificar la apertura exitosa de archivos y gestionar errores de manera adecuada para garantizar la estabilidad del programa.

Posicionamiento en Archivos:

Los archivos de texto pueden ser extensos, y es fundamental entender cómo posicionarse en ellos. Los punteros de posición, como seekg y seekp, permiten navegar a ubicaciones específicas dentro de un archivo, facilitando la lectura o escritura en puntos particulares.

Formato y Manipulación de Datos:

C++ proporciona diversas funciones y manipuladores para formatear la salida y entrada de datos en archivos. Estos incluyen manipuladores como setw para establecer la anchura del campo y setprecision para controlar la precisión de los números de punto flotante. La comprensión de estas herramientas es esencial para presentar y procesar datos de manera legible y precisa.

Archivos Binarios vs. Archivos de Texto:

Aunque nos estamos centrando en archivos de texto, es crucial entender la diferencia entre archivos binarios y de texto. Los archivos binarios contienen información en un formato más compacto y suelen ser más eficientes, pero los archivos de texto son más legibles y pueden ser editados fácilmente con un editor de texto simple. La elección entre ellos depende de los requisitos específicos del programa.

Manejo de Fin de Archivo (EOF):

El manejo adecuado de la condición de fin de archivo es esencial para evitar errores en la lectura de archivos. La función eof() permite verificar si se ha alcanzado el final del archivo, evitando la lectura innecesaria y asegurando un procesamiento preciso de los datos.

Operaciones de Manipulación de Archivos:

Además de la lectura y escritura básicas, C++ ofrece operaciones como la eliminación de archivos (remove) y la comprobación de existencia (ifstream::good()) para garantizar un manejo integral de los archivos. Estas operaciones son esenciales para el mantenimiento y la gestión adecuada de archivos en un sistema.

Serialización y Deserialización:

En muchos casos, se requiere la serialización de datos para almacenar objetos complejos en archivos y su posterior deserialización para recuperar la estructura original. El conocimiento de estas técnicas es fundamental para el manejo avanzado de archivos y el almacenamiento persistente de datos complejos.

Procedimiento:

ACTIVIDAD 12
Archivos Texto

MENÚ

1.- Cargar Archivo
2.- Agregar
3.- Eliminar
4.- Buscar
5.- Ordenar
6.- Mostrar Todo
7.- Generar Archivo
0.- Salir

INSTRUCCIONES: Programa que contenga el menú anterior, el programa utiliza un vector de registros de máximo 1,500 registros, de la siguiente estructura: [status, matricula, ApPat, ApMat, Nombre, Edad, sexo] donde el campo llave es matricula.
datos.txt es el archivo con los registros a cargar en el vector de registros

1.- Cargar Archivo : El programa deberá cargar el vector de registros desde el archivo de texto **(solo podrá cargarse una sola vez el archivo)**

2.- Agregar : El programa deberá ser capaz de agregar un 10 registros al arreglo y al final del archivo de texto. **(Generar automáticamente los datos)**.

3.- Buscar : El programa deberá buscar una matrícula en el vector por medio del método de búsqueda más óptimo. Utilizar banderas para escoger el método más adecuado.

4.- Ordenar : El programa deberá ordenar el vector por medio del método de ordenación más óptimo. Utilizar banderas para escoger el método más adecuado se ordenará por el **campo llave (matrícula)**

5.- Mostrar Todo: El programa deberá mostrar todos los registros del vector tal y como están en ese momento ordenado o desordenado.

6.- Generar Archivo : El programa deberá preguntar al usuario el nombre del archivo, **solo nombre sin extensión**, el programa generará un archivo con el nombre proporcionado por el usuario con **extensión .txt** los datos que pondrá en el archivo de texto serán idénticos a los contenidos en el Vector de registros. (ordenado o desordenado). El programa podrá generar múltiples archivos para comprobar las salidas.

NOTA: Programa 100% Validado.
NOTA 2: Si el programa vector cambia de tamaño por agregar un nuevo registro al salir el programa deberá actualizar o sustituir el **datos.txt**
NOTA 3: Usar libreria propia
NOTA 4: archivo.txt es un ejemplo de cómo debe ser el archivo que se genere

Conclusiones:

El manejo de archivos de texto en C++ es una habilidad fundamental para cualquier programador, ya que proporciona una forma eficaz de almacenar, procesar y recuperar datos de manera persistente. La competencia en este campo no solo implica conocimientos técnicos, sino también la capacidad de implementar soluciones eficientes y seguras.

Anexo:

https://github.com/BrayanArM98/Estructurada