UNIVERSIDAD AUTÓNOMA DE BAJA CALIFORNIA

Ingeniería en Computación

Actividad 14 Archivos indexados

Programación Estructurada

***

ALUMNO: Arredondo Urbalejo Isai Alexis

MATRÍCULA: 368747

GRUPO: 932

PROFESOR: Pedro Nunez Yepiz

Introducción

Los archivos indexados son una estructura de datos que se utiliza para almacenar información de forma organizada y eficiente. En el caso de los archivos indexados en C++, se utiliza una estructura de datos llamada índice para almacenar la ubicación de los datos en el archivo.

Los índices permiten realizar búsquedas de datos de forma rápida y eficiente. Por ejemplo, si tenemos un archivo de texto con una lista de nombres y apellidos, podemos utilizar un índice para buscar un nombre específico. El índice nos indicará la ubicación del nombre en el archivo, lo que nos permitirá acceder a él de forma directa.

Competencia

Comprender los conceptos básicos de los archivos indexados en C++. Conocer los diferentes tipos de índices que existen. Analizar las ventajas y desventajas de cada tipo de índice. Aprender a crear y utilizar índices en C++.

Fundamentos

Los archivos indexados en C++ son una estructura de datos compuesta por dos componentes:

- Archivo de datos: Este archivo contiene los datos que se desean almacenar.
- Índice: Este archivo contiene la ubicación de los datos en el archivo de datos.

El índice se suele almacenar en un archivo separado del archivo de datos. Esto permite que el índice sea más pequeño y eficiente, ya que solo almacena la información necesaria para realizar búsquedas.

Tipos de índices

Existen diferentes tipos de índices que se pueden utilizar en C++. Los principales tipos de índices son los siguientes:

- Índices secuenciales: Los índices secuenciales almacenan la ubicación de los datos en orden secuencial. Este tipo de índice es sencillo de implementar, pero no es muy eficiente para búsquedas no secuenciales.
- Índices binarios: Los índices binarios almacenan la ubicación de los datos en un árbol binario. Este tipo de índice es más eficiente que los índices secuenciales para búsquedas no secuenciales, pero es más complejo de implementar.
- Índices hash: Los índices hash almacenan la ubicación de los datos en una tabla hash. Este tipo de índice es muy eficiente para búsquedas, pero puede ser menos eficiente para inserciones y eliminaciones.

Ventajas y desventajas de los índices

Los índices ofrecen las siguientes ventajas:

- Permiten realizar búsquedas de datos de forma rápida y eficiente.
- Reducen la cantidad de datos que deben ser leídos del archivo de datos.
- Mejoran el rendimiento de las aplicaciones que realizan operaciones de búsqueda.

Los índices también tienen las siguientes desventajas:

- Aumentan el tamaño del archivo de datos.
- Pueden ser complejos de implementar.
- Pueden reducir la eficiencia de las operaciones de inserción y eliminación.

Creación y utilización de índices

Para crear un índice en C++, se pueden utilizar las bibliotecas estándar de C++ o bibliotecas de terceros. Las bibliotecas estándar de C++ proporcionan funciones para crear índices secuenciales y binarios. Las bibliotecas de terceros pueden proporcionar funciones para crear otros tipos de índices, como índices hash.

Para utilizar un índice en C++, se pueden utilizar las funciones proporcionadas por la biblioteca estándar o la biblioteca de terceros. Las funciones proporcionadas por la biblioteca estándar permiten buscar datos en un índice de forma secuencial o binaria. Las funciones proporcionadas por la biblioteca de terceros pueden permitir buscar datos en un índice de forma más eficiente, utilizando algoritmos específicos para el tipo de índice utilizado.

Resumen: ARCHIVOS INDEXADOS | Algoritmo y Estructura de Datos | Ingeniería en Sistemas de Información (UTN) | | Filadd. (s. f.). https://filadd.com/doc/archivo-indexado-docx-algoritmo-y-estructura-de

Archivos secuenciales-indexados C++. (s. f.). PPT. https://es.slideshare.net/EdsonRc/archivos-secuencialesindexados-c

ACTIVIDAD 14

Archivos Binarios

(archivos indexados)

MENÚ

1.- AGREGAR
2.- ELIMINAR
3.- BUSCAR
4.- ORDENAR
5.- IMPRIMIR REGISTROS ARCHIVO ORIGINAL
6.- IMPRIMIR REGISTROS ARCHIVO ORDENADO
7.- GENERAR ARCHIVO TEXTO
8.- EMPAQUETAR
0.- SALIR

INSTRUCCIONES: Programa que contenga el menú anterior, el programa utiliza un vector de índices de la siguiente estructura: **[llave, índice]** donde *el campo llave es* **noempleado.**
registros.dat es el archivo con los registros a cargar en el vector de índices **archivo binario sera proporcionado**,

CARGAR ARCHIVO : El programa deberá cargar al arrancar el programa, el archivo Binario generará el vector de índices (**llave, indice**) sólo con registros válidos **(el tamaño del vector debera ser 25% mas grande que el la cantidad de registros que contenga el archivo binario) utiliza un archivo externo para averiguar tamaño y retorne cantidad de registros.**

1.- Agregar :
El programa deberá ser capaz de agregar un registro al arreglo de índices y al final del archivo Binario. (agregar forma automatica no repetido el campo llave)

2.- Eliminar :

- El programa deberá buscar una **noempleado** en el vector de índices por medio del método de búsqueda más óptimo.
- La función deberá *retornar,* el índice donde se encuentra la matrícula en el archivo Binario, utilizar banderas para escoger el método más adecuado.

3.- Buscar :

- El programa deberá buscar un **noempleado** en el vector de índices por medio del método de búsqueda más óptimo.
- La función deberá *retornar,* el índice donde se encuentra la matrícula en el archivo Binario, utilizar banderas para escoger el método más adecuado.
- Una vez obtenido el índice moverse dentro del archivo binario (usar fseek) usando el índice del vector de índices.
- Leer el registro en la posición correcta, y desplegar el registro.

4.- Ordenar :

El programa deberá ordenar el vector de índices por medio del método de ordenación más óptimo. Utilizar banderas para escoger el método más adecuado por el que se ordenará por el campo llave (**noempleado**) o no ordenarse si ya está ordenado. (utilizar 3 metodos de ordenacion diferentes segun sea el caso que se necesite Justificar los metodos en el reporte)

5 Y 6.- Mostrar Todo:

El programa deberá mostrar todos los registros del Archivo Binario, preguntar: **ordenado o normal**. Usar el vector de índices para imprimirlo ordenado, y directamente desde el archivo si es normal.

7.- GENERAR ARCHIVO TEXTO:

El programa deberá generar un archivo de texto, el usuario debe proporcionar el nombre del archivo.
El programa deberá mostrar todos los registros del Archivo Binario, preguntar: **ordenado o normal**. Usar el vector de índices para imprimirlo ordenado, y directamente desde el archivo si es normal.
el programa podrá generar múltiples archivos para comprobar las salidas.

8.- EMPAQUETAR :

El programa deberá actualizar el Archivo Binario, a partir de solo registros válidos, y eliminarlos del archivo binario. Crear copia y archivo de respaldo .bak del archivo de antes de eliminarlos.

Resultados y conclusiones

En conclusión, los archivos indexados son una herramienta valiosa para el desarrollo de aplicaciones en C++. Permiten realizar búsquedas de datos de forma rápida y

eficiente, mejorando el rendimiento de las aplicaciones que realizan operaciones de búsqueda.

La elección del tipo de índice adecuado depende de las necesidades específicas de la aplicación. Si las búsquedas son secuenciales, los índices secuenciales pueden ser una buena opción. Si las búsquedas son no secuenciales, los índices binarios o hash pueden ser una mejor opción.

Los índices también pueden aumentar el tamaño del archivo de datos y reducir la eficiencia de las operaciones de inserción y eliminación. Por lo tanto, es importante evaluar cuidadosamente las ventajas y desventajas de los índices antes de utilizarlos en una aplicación.