Universidad Rafael Landívar

Facultad de Ingeniería

Campus Central

Física I

Angel Daniel Avila Arriaza

Carnet: 1180622

Programación Avanzada Segundo Ciclo 2023

Proyecto de aplicación 2

Carnet: 1180622

November 2023

1 Introduction

Abstract

En el ámbito actual de las organizaciones, la gestión eficiente de datos se ha convertido en un pilar fundamental para el desarrollo y funcionamiento efectivo de aplicativos y sistemas. En este contexto, se presenta un proyecto que aborda la creación de un programa en C++ diseñado para administrar grandes volúmenes de datos en memoria. Este programa no solo se centra en la carga y ordenamiento de información, sino también en la actualización de datos, reconociendo la importancia crítica de mantener la integridad y disponibilidad de la información.

La premisa central del proyecto radica en la creación de un sistema capaz de manejar registros únicos, representados por valores alfanuméricos, los cuales se cargan desde archivos de texto. Estos archivos siguen un formato específico, donde cada línea constituye un conjunto de datos, separados por comas. La primera y última entrada de cada conjunto actúan como claves para la identificación y relación entre elementos, introduciendo la necesidad de implementar una función hash que garantice la unicidad de las claves.

El sistema propuesto inicia solicitando al usuario la ruta o nombre del archivo que contiene los conjuntos de datos. Posteriormente, mediante una función hash, el programa asigna una clave única de 10 caracteres a cada conjunto de datos, permitiendo su identificación en una lista doblemente enlazada. La carga de datos se realiza de forma ordenada, ascendente, respecto a la clave hash.

Una vez en memoria, el programa ofrece tres funcionalidades clave al usuario. En primer lugar, la posibilidad de cargar más datos desde archivos de texto, manteniendo la coherencia con el formato establecido. En segundo lugar, la búsqueda de datos por clave, implementada a través de un eficiente algoritmo de búsqueda binaria. Esta búsqueda implica la aplicación de la función hash para comparar las claves almacenadas en el programa y mostrar los conjuntos de datos asociados a la clave proporcionada por el usuario.

La tercera opción se centra en la búsqueda de datos por valor, donde el usuario ingresa un término y el programa busca entre los conjuntos de datos aquellos que contienen un dato coincidente con el término buscado. Esta búsqueda se lleva a cabo de manera secuencial en los conjuntos de datos y, opcionalmente, se puede optimizar mediante una búsqueda binaria interna, aprovechando el ordenamiento de los datos en memoria.

2 Analisis del problema

El programa debe comenzar solicitando al usuario la ruta o nombre del archivo que contiene los conjuntos de datos. Posteriormente, debe cargar toda la información del archivo en memoria. Se asume que el archivo no contiene errores, pero puede contener conjuntos duplicados. La carga de datos implica la aplicación de una función hash a la clave primaria de cada conjunto, generando una clave única de 10 caracteres que servirá para identificar el registro en una lista doblemente enlazada. La inserción de datos en la lista se realiza de manera ordenada, ascendente, respecto a la clave hash.

Una vez que el programa tiene los datos en memoria, debe ofrecer tres funcionalidades al usuario. En primer lugar, la posibilidad de cargar más datos desde archivos de texto, manteniendo la coherencia con el formato establecido. En segundo lugar, la búsqueda de datos por clave, implementada mediante una búsqueda binaria. Esta búsqueda implica la aplicación de la función hash para comparar las claves almacenadas y mostrar los conjuntos de datos asociados a la clave proporcionada por el usuario.

La tercera opción se centra en la búsqueda de datos por valor, donde el usuario ingresa un término y el programa busca entre los conjuntos de datos aquellos que contienen un dato coincidente con el término buscado. Esta búsqueda se realiza de manera secuencial en los conjuntos de datos y, opcionalmente, se puede optimizar mediante una búsqueda binaria interna, aprovechando el ordenamiento de los datos en memoria.]El problema planteado se enfoca en el diseño y desarrollo de un programa en lenguaje de programación C++ para la gestión eficiente de grandes cantidades de datos en memoria. El objetivo principal es crear una solución que permita la carga, ordenamiento y actualización de registros únicos representados por valores alfanuméricos. Estos registros se obtienen desde archivos de texto que siguen un formato específico, donde cada línea constituye un conjunto de datos separados por comas. La primera y última entrada de cada conjunto actúan como claves para identificación y relación entre elementos.

El programa debe comenzar solicitando al usuario la ruta o nombre del archivo que contiene los conjuntos de datos. Posteriormente, debe cargar toda la información del archivo en memoria. Se asume que el archivo no contiene errores, pero puede contener conjuntos duplicados. La carga de datos implica la aplicación de una función hash a la clave primaria de cada conjunto, generando una clave única de 10 caracteres que servirá para identificar el registro en una lista doblemente enlazada. La inserción de datos en la lista se realiza de manera ordenada, ascendente, respecto a la clave hash.

Una vez que el programa tiene los datos en memoria, debe ofrecer tres funcionalidades al usuario. En primer lugar, la posibilidad de cargar más datos desde archivos de texto, manteniendo la coherencia con el formato establecido. En segundo lugar, la búsqueda de datos por clave, implementada mediante una búsqueda binaria. Esta búsqueda implica la aplicación de la función hash para comparar las claves almacenadas y mostrar los conjuntos de datos asociados a la clave proporcionada por el usuario.

La tercera opción se centra en la búsqueda de datos por valor, donde el usuario ingresa un término y el programa busca entre los conjuntos de datos aquellos que contienen un dato coincidente con el término buscado. Esta búsqueda se realiza de manera secuencial en los conjuntos de datos y, opcionalmente, se puede optimizar mediante una búsqueda binaria interna, aprovechando el ordenamiento de los datos en memoria.

3 Marco Teorico

La gestión eficiente de datos constituye un componente fundamental en el ámbito de la informática y la ingeniería de software. En el contexto de este proyecto, se abordan diversos conceptos y técnicas que forman parte del marco teórico necesario para comprender y resolver los desafíos asociados con la administración de grandes cantidades de datos en memoria utilizando el lenguaje de programación C++.

1. Funciones Hash:

Las funciones hash desempeñan un papel crucial en la garantía de la unicidad y la eficiencia en la gestión de datos. Una función hash toma una entrada, en este caso, la clave primaria de un conjunto de datos, y devuelve una cadena de longitud fija que actúa como identificador único. La propiedad esencial de estas funciones es que cambios mínimos en la entrada deben generar cambios significativos en la salida, proporcionando así una distribución uniforme de las claves hash.

En el contexto de este proyecto, la función hash se utiliza para asignar claves únicas de 10 caracteres a cada conjunto de datos. Esta asignación permite la identificación eficiente de registros en una lista doblemente enlazada, facilitando operaciones como la búsqueda y actualización.

2. Listas Doblemente Enlazadas:

Las listas doblemente enlazadas son estructuras de datos que permiten un acceso eficiente tanto hacia adelante como hacia atrás. Cada nodo en la lista contiene datos y referencias a los nodos vecinos, lo que facilita la inserción y eliminación de elementos en cualquier posición. En el contexto de este proyecto, las listas doblemente enlazadas se utilizan para almacenar los conjuntos de datos de manera ordenada según sus claves hash.

La elección de esta estructura de datos se basa en su capacidad para mantener la coherencia en la secuencia de datos, facilitando la implementación de búsquedas eficientes y la actualización de información.

3. Búsqueda Binaria:

La búsqueda binaria es un algoritmo eficiente para encontrar un elemento en una lista ordenada. Este algoritmo divide repetidamente la lista a la mitad, descartando la mitad en la que no se encuentra el elemento deseado. La búsqueda binaria es especialmente útil cuando se trata de conjuntos de datos ordenados, ya que reduce significativamente el número de comparaciones necesarias para encontrar un elemento.

En el proyecto, la búsqueda binaria se emplea en dos contextos distintos: primero, para buscar conjuntos de datos por clave y, segundo, para optimizar

la búsqueda dentro de los conjuntos de datos por valor.

4. Manejo de Archivos de Texto en C++:

La manipulación de archivos en C++ es esencial para la carga y almacenamiento de datos. La biblioteca estándar de C++ proporciona clases y funciones para realizar operaciones de lectura y escritura en archivos de texto. En este proyecto, la capacidad de leer datos desde archivos de texto y cargarlos en memoria es un componente clave para la funcionalidad del sistema.

4 Conclusiones

El marco teórico abordado destaca la importancia de conceptos como funciones hash, listas doblemente enlazadas, búsqueda binaria y manejo de archivos en la resolución de problemas relacionados con la administración de datos. La combinación adecuada de estos elementos permite diseñar un programa eficiente y robusto capaz de abordar los desafíos específicos de este proyecto. La comprensión profunda de estos conceptos proporciona las bases teóricas necesarias para la implementación exitosa del sistema propuesto.

5 Bibliografia

https://en.wikipedia.org/wiki/Hash$_f unctionhttps : //www.geeksforgeeks.org/hashing-data-structure/https : //en.wikipedia.org/wiki/Doubly_linked_listhttps : //www.geeksforgeeks.org/doubly-linked-list/https : //en.wikipedia.org/wiki/Binary_search_algorithm$