Manual Tecnico

Gestor de Contactos en Consola C++

Nombre

> Maria Monserrat Gomez Rabatu

Carnet

> 202030849

Curso

> Laboratorio Estructura de Datos

< PRIMER SEMESTRE 2024 >

Requisitos del Sistema

Para utilizar o modificar el gestor de contactos en consola, tu sistema debe cumplir con los siguientes requisitos:

- **Sistema Operativo:** Windows 7 o superior, macOS, o una distribución de Linux compatible.
- **Compilador de C++:** Es necesario tener instalado un compilador de C++ como GCC, Clang o MSVC. Asegúrate de que el compilador esté correctamente configurado en el PATH del sistema para poder invocarlo desde la consola.
- **Librerías Estándar de C++:** Tu sistema debe tener las librerías estándar de C++ disponibles para el compilador.
- **Espacio en Disco:** Al menos 100 MB de espacio libre en disco para la instalación y almacenamiento de datos.
- **Memoria RAM:** Se recomienda tener al menos 512 MB de memoria RAM para un rendimiento óptimo.
- **IDE recomendado:** CLion.

◊ Instalación de un Compilador de C++

a. Windows:

Visual Studio: Instala Visual Studio y selecciona el componente "Desarrollo para escritorio con C++" durante la instalación.

MinGW: Descarga e instala MinGW para obtener GCC.

b. macOS:

Utiliza el comando xcode-select --install en la terminal para instalar las herramientas de línea de comandos de Xcode, que incluyen Clang.

c. Linux:

La mayoría de las distribuciones de Linux vienen con GCC instalado. Si no es así, puedes instalarlo usando el gestor de paquetes de tu distribución, por ejemplo, sudo apt install build-essential en distribuciones basadas en Debian.

Introducción

El Gestor de Contactos en Consola es una implementación del una agenda diseñado para ser ejecutado en una interfaz de línea de comandos. Este manual proporciona una visión general de la estructura y el funcionamiento del programa, así como una guía para su uso y desarrollo.

Estructura del Proyecto

El proyecto está organizado en varias clases, cada una con un propósito específico:

- **main.cpp:** Contiene la función principal del programa.
- **AVLNode.cpp/h:** Define la clase Nodo para un Árbol AVL, contiene un contacto y dos punteros (left, right).
- **AVLTree.cpp/h:** Implementa la estructura de un Árbol AVL.
- **ComandoManager.cpp/h:** Maneja todo lo relacionado a la linea de commandos de la terminal.
- **Contacto.cpp/h:** Define la clase Contacto, representa un contacto con 3 parametros (nombre, apellido, num).
- **ExportManager.cpp/h:** Maneja todo lo relacionado a las acciones para exportar.
- **GrupoContactos.cpp/h:** Define la clase de grupo de contactos, que contiene un árbol AVL y su nombre.
- **HashTable.cpp/h:** Implementa la estructura de una table Hash.
- **ReporteManager.cpp/h:** Maneja todo lo relacionado a las acciones de los reportes.

Clase AVLNode:

La clase AVLNODE representa un nodo para la estructura de árbol AVL, tiene los siguientes atributos y métodos:

- **Contacto data:** Es un valor Contacto
- **AVLNode * left:** Es un puntero a un nodo AVL, representa un hijo al lado izquierdo.
- **AVLNode * right:** Es un puntero a un nodo AVL, representa un hijo al lado derecho.
- **Int height:** Es la altura en el árbol donde esta colocado el nodo.

* **Constructor:** Se necesita un contacto para crear un nodo AVL, el int height predefinido es 1 y los punteros left/right apuntan a nullptr.

Clase AVLTree:

La clase AVLTREE representa la implementacion de un árbol AVL, tiene los siguientes atributos y métodos:

- **AVLNode * raiz:** Es el valor del nodo AVL que inicia el árbol.

* **Constructor:**
  Se necesita un contacto para crear un nodo AVL, el int height predefinido es 1 y los punteros left/right apuntan a nullptr.

* **List<Contacto> buscarPorNum, buscarPorNombre, buscarPorApellido:**
  Metodos para recorrer el arbol AVL y encontrar contactos segun el parametro requerido, regresa una lista con los resultados.

* **AVLNode * rotateRight, rotateLeft:** Metodos auxiliares para balancear el arbol AVL

* **AVLNode * insertHelper:**
  Metodo para agregar un nodo al árbol AVL, se mueve a través del arbol por medio de los datos del contacto en el nodo.

* **Void exportHelper:**
  Conecta con la clase ExportManager para exportar los contactos del arbol a un archivo txt por cada contacto.

* Los demas metodos son auxiliares para empezar los recorridos necesarios, o para obtener datos del arbol.

Clase ComandoManager:

La clase COMANDOMANAGER se encarga de administrar los comandos que se ingresan en la terminal del gestor.

* **string limpiarValor:** Metodo interno auxiliar para eliminar caracteres innecesarios de los valores.

* **void procesarComandoBusqueda, procesarComandoAdd:**
  Metodo que analiza la entrada en la terminal, de termina que tenga los parametros necesarios y realiza lo solicitado (busqueda o agregar, los metodos agregarIndividual y agregarGrupo son sub metodos del metodo procesarComandoAdd).

* **Void agregarIndividual:**
  Analiza los parametros en la entrada de la terminal, crea un nuevo contacto y lo agrega al grupo indicado.

* **void agregarGrupo:**
  Analiza los parametros en la entrada de la terminal, crea un grupo nuevo, el cual necesita al menos un contacto.

Clase Contacto:

La clase CONTACTO representa un objeto de Contacto, teniendo los siguientes atributos:

- **string nombre:** El nombre asignado al contacto.
- **string lastname:** El apellido asignado al contacto.
- **int numero:** El numero asignado al contacto

◊ **Constructor vacio:** Constructor para crear un contacto por defecto

◊ **Constructor:** Se necesita de los 3 parametros anteriores para crearlo.

◊ Los demas metodos son auxiliares acceder a los atributos privados del objeto (get y set).

Clase ExportManager:

La clase EXPORTMANAGER se encarga de administrar la accion de exportar uno o varios contactos y el log.

◊ **void exportarNuevoDoc:**
Metodo para crear un archivo (si no existe) y escribir los datos del contacto en un archivo txt.

◊ **void exportarDocGeneral:**
Metodo para actualizar el documento general utilizado para importar los contactos, utiliza el metodo "exportarNuevoDoc" con la direccion del archivo txt, asi sobreescribiendolo.

◊ **void exportarLog:** Metodo para crear o actualizar el archivo log del sistema.

Clase GrupoContactos:

La clase GRUPOCONTACTOS representa un objeto de un grupo de contactos, que tiene los siguientes atributos:

- **string nombre:** El nombre asignado al grupo.
- **AVLTree contactos:** El arbol AVL que administra los contactos a traves de sus nodos.

◊ **Constructor:** Se necesita unicamente el atributo nombre para crear un grupo.

Clase HashTable:

La clase HASHTABLE representa la implementacion de una tabla Hash, con los siguientes metodos:

- **struct HashNode:**
  Nodo utilizado para evitar colisiones en la tabla, contiene un GrupoContacto * grupo y un HashNode * siguiente, funcionando como una lista enlazada.
- **HashNode * table[cantidad]:** Array de HashNode, vease tambien como la tabla Hash.

◊ **int funcionHash:** Metodo interno para asignar lugar en la tabla, por medio de una llave string.

◊ **Constructor:** Asigna a cada posicion de la tabla un valor nullptr.

◊ **Desctructor:** Asigna a cada posicion de la tabla un valor nullprt.

◊ **void agregarGrupo:** Metodo para agregar un grupo de contactos a la tabla.

◊ **GrupoContactos * buscarGrupo:**
Recorre la tabla (y los nodos en ella de ser necesario) para buscar un grupo.

◊ Los demas metodos son auxiliares acceder a los atributos privados del objeto (get y set).

Clase ExportManager:

La clase EXPORTMANAGER se encarga de administrar la accion de exportar uno o varios contactos y el log.

◊ **void exportarNuevoDoc:**
Metodo para crear un archivo (si no existe) y escribir los datos del contacto en un archivo txt.

◊ **void exportarDocGeneral:** Metodo para actualizar el documento general utilizado para importar los contactos, utiliza el metodo "exportarNuevoDoc" con la direccion del archivo txt, asi sobreescribiendolo.

◊ **void exportarLog:** Metodo para crear o actualizar el archivo log del sistema.

Clase GrupoContactos:

La clase GRUPOCONTACTOS representa un objeto de un grupo de contactos, que tiene los siguientes atributos:

- **string nombre:** El nombre asignado al grupo.
- **AVLTree contactos:** El arbol AVL que administra los contactos a traves de sus nodos.

◊ **Constructor:** Se necesita unicamente el atributo nombre para crear un grupo.

Clase ReporteManager:

La clase REPORTEMANAGER agrupa los metodos utilizados para la realizacion de reportes, con los atributos:

- **string log:**
  Guarda cada acción realizada en el sistema, ya sea una busqueda, agregar un contacto o un grupo, y la actualizacion del documento utilizado para importar los contactos ya existentes.

◊ **void datosSistemas:** Muestra todos los datos guardados en el sistema.

◊ **void logSistema:**
Agrega al string log una nueva accion, agregando su indicador, y la fecha y hora en la que se realizaron las acciones.

◊ **void mostrarLog:** Muestra los datos guardados en el log.

◊ **void exportarLog:**
Exporta a través de ExportManager el log del sistema en un archivo txt en los archivos del proyecto.

◊ **void datosPorGrupo:** Metodo que muestra cuantos contactos hay en un grupo determinado.

Uso del Programa:

Para ejecutar el Gestor de Contactos en Consola, simplemente compila con el archivo makefile, y se puede proceder a ejecutar el archivo exe resultante

Desarrollo Adicional:

El código fuente proporcionado puede ser extendido y modificado según las necesidades del usuario. Se recomienda tener un conocimiento básico de C++ y estructuras de datos para realizar cambios significativos en el programa.

Ambiente de Desarrollo:

El desarrollo de este proyecto se llevó a cabo en Windows 11 utilizando el IDE de CLion, durante el mes de marzo de 2024.

Contacto:

Para cualquier consulta o reporte de problemas relacionados con el Gestor de Contactos en Consola, por favor contacta al desarrollador en mgomezrabatu@gmail.com