Instituto Tecnológico de Costa Rica

Sede Interuniversitaria

Ingeniería en Computación

Estructuras de Datos - II Proyecto Programado

Profesora: Samanta Ramijan Carmiol

Estudiantes: Natán Fernández de Castro - 2017105774
Pablo Venegas Sánchez - 2018083497

Documentación Externa

PASS-TEC Organizador de Contraseñas en Lenguaje C

II Semestre 2018

1. Tabla de Contenidos

2. Descripción del Problema

El problema consiste en implementar un programa Administrador de contraseñas. Este programa debe ser capaz de generar una contraseña según los parámetros que decida el

usuario. Los cuales serían el largo de la contraseña a generar, si desea incluir mayúsculas o símbolos. Luego ingresa la plataforma y el usuario relacionados con la contraseña. Esto debe ingresarse como un nodo en un árbol Splay. El usuario tiene la posibilidad además de borrar, obtener la contraseña, además de mostrar todo el contenido del árbol binario mediante recorrido Preorden. Otro aspecto sería la posibilidad de cargar y guardar el estado de un árbol binario generado mediante manejo de archivo .txt.

3. Diseño del Programa

3.1 Decisiones de diseño

Se decidió implementar patrón de diseño Modelo Vista Controlador, en cual permite asignar responsabilidades en lo que respecta a eventos, peticiones y obtención de datos. Es decir, se divide la parte visual, datos (lógica) y obtención de esos datos. Esto permitiría mayor orden en la estructuración del diseño del programa, corrección de errores y control sobre estructuras.

Además se decide por medio de archivos de cabecera (.h) controlar las funciones privadas y públicas que cada archivo .c maneja. Estos permite controlar la información que se muestra sobre la solución e implementación.

Para la generación de la contraseña se decide utilizar un alfabeto según las opciones del usuario con respecto a la inclusión de mayúsculas, símbolos o ninguno de los dos. Estas preferencias variaría el alfabeto para cada contraseña. Teniendo el alfabeto generado mediante la utilización de la biblioteca rand(), propia de lenguaje C, se genera un número aleatorio en los rangos del largo del alfabeto que vendría a ser un caracter aleatorio en esa posición del entero generado. Ahora, se va llenando el char de contraseña hasta que sea igual que el largo que el usuario desea la contraseña. De esta manera tenemos como resultado una contraseña generada.

Luego se le solicita al usuario la información que corresponde con la contraseña tales como: Plataforma y usuario. Se procede luego de esto al manejo de estructura tipo Información en el que transferiremos la información ingresada y generada. La cual se insertará en un árbol binario realizando biselaciones, con el propósito de mantener las contraseñas frecuentes en la parte superior del árbol binario. Estas biselaciones se realizarán cada vez que el usuario decida obtener una contraseña (Dado que cuando la contraseña se genera no se le muestra, solo se guarda) y borrar una contraseña. Estos tres casos utilizarán la misma función de biselación en la cual se rotará analizando el estado del árbol y donde se encuentre el árbol binario según su estado de manera que el nodo manipulado o nuevo ingrese a la raíz.

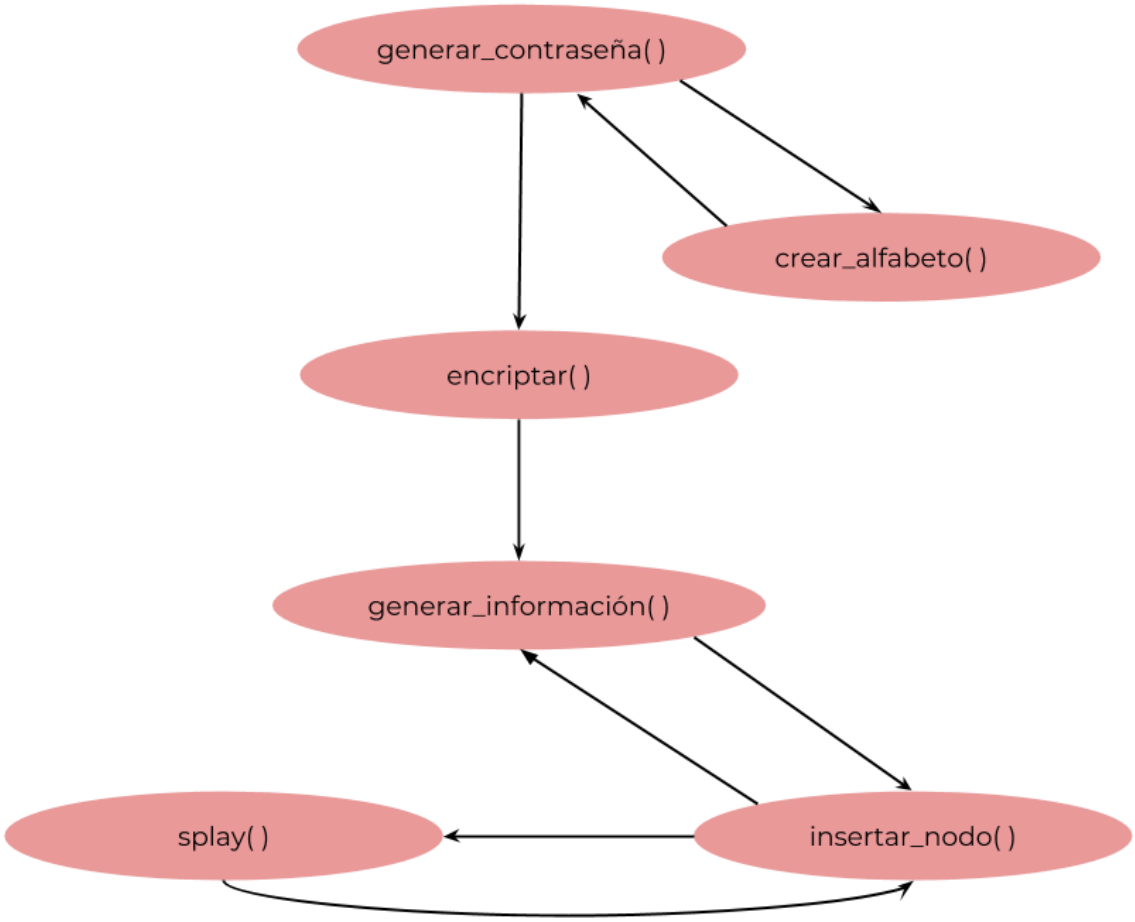
Cuando el usuario desee que se le muestre la información que ha ingresado (contenido del árbol binario) se realizará un recorrido Preorden del árbol binario.

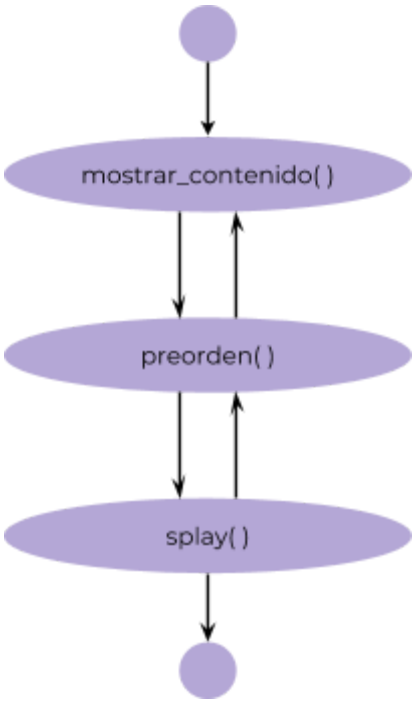
Algoritmos utilizados:

- **generar_contrasena:** Realiza llamada función crear_alfabeto(), con el alfabeto creado para la generación de la contraseña se le pregunta al usuario el largo de la contraseña que desea, se captura y se crean dos char, uno pass que el tamaño va a ser el mismo que el largo deseado. Se utiliza la biblioteca rand(), por lo que se inicia la semilla utilizada para generar un número aleatorio mediante srand(), se itera sobre un char temporal concatenando un caracter en una posición aleatoria (mediante el número aleatorio generado) del alfabeto. Al ser el índice de la iteración igual al largo de contraseña deseado se concatena el resultado al char pass. Por último se llama a una función de encripción para mayor seguridad de almacenamiento de la contraseña pasando por parámetro el char pass.
- **crear_alfabeto:** Esta función se basa principalmente en validaciones según los resultados de captura si el usuario desea mayúsculas o símbolos. Se pueden dar cuatro bifurcaciones: mayúsculas y símbolos ambos con opción sí, mayúsculas pero no símbolos, símbolos pero no mayúsculas y ambos con opción no. Estos se transfieren a una variable global de alfabeto que se guardará en el nodo futuro de información para saber el alfabeto respectivo de cada contraseña.
- **encriptar:** Manipula el char de contraseña generado con el propósito de encriptarlo mediante cifrado simétrico AES el cual utiliza llave de 256 bit y un vector de inicialización de 128 bit. El tamaño de la llave y del vector de inicialización le brindan a la contraseña para almacenar cifrado de seguridad pesada. Se implementa biblioteca OpenSSL de encripción-desencripción. Luego de encriptar la contraseña, los valores resultantes se pasan por parámetro a la función generar_informacion().
- **generar_informacion:** Captura la información que corresponde a la contraseña generada, tal como: Plataforma y usuario. Se le asigna un identificador a la información que va a ser utilizada luego para acceder a la información una vez que esté presente en el árbol binario. Además inserta la información en un nuevo nodo de tipo de tipo Información. y luego actualiza el árbol binario llamando a la función insertar_nodo() con el árbol binario y la nueva información como parámetro.
- **insertar_nodo:** Analiza si el árbol no posee nodo en la raíz, de ser así simplemente inserta el nodo en la raíz. De no ser así el nodo raíz va a ser igual a una llamada a la función splay(), la cual va a llevar el nodo nuevo a la raíz del árbol binario mediante rotaciones zig y zag.

- **mostrar_contenido:** Función en el caso que el usuario desee ver el contenido del árbol binario. Se llama a la función preOrden() con el árbol binario como parámetro.
- **preOrden:** Valida si el árbol binario no está vacío, caso que sea verdadero muestra la información del nodo y luego se llama por recursión la función con el hijo izquierdo como raíz, y luego se llama con el hijo derecho como raíz, es decir una vez que imprima por profundidad todo el subárbol izquierdo a partir de la raíz, muestra el subárbol derecho.
- **obtener_contrasena:** Captura el identificador de la contraseña que el usuario desea obtener, e iguala el árbol binario a la llamada de la función buscar_contraseña con el árbol binario y el identificador a buscar como parámetro. Una vez retornado la biselación muestra la raíz del árbol como la contraseña encontrada.
- **buscar_contraseña:** Llama a función splay() que bisela el árbol en busca de el identificador de la contraseña.
- **borrar_contraseña:** Captura el identificador el cual el usuario desea borrar la contraseña, iguala el árbol binario a la llamada de función biselar. Caso exitoso que borre la contraseña imprime mensaje y en caso de no encontrar imprime error.

Diagramas de flujo:





4. Análisis de Resultados

Objetivos Alcanzados

- Los resultados del trabajo satisfactorios, se lograron implementar los requerimientos de manera exitosa y funcional.
- Mejor proceso de trabajo al trabajar en un solo sistema operativo.
- Lograr la tarea extra de encriptar y desencriptar la contraseña almacenada.

5. Manual del Usuario

generar_contrasena: Comando en el cual genera una nueva contraseña, la cifra y la inserta biselando en el árbol .
mostrar_contenido: Comando el cual muestra el contenido actual del árbol binario mediante recorrido preorden.
obtener_contrasena: Comando el cual le muestra al usuario la contraseña según el identificador que ingresó.
guardar_archivo: Guarda en un archivo .txt el estado actual del árbol binario.
cargar_archivo: Cargar estado previo del sistema en archivo de texto .txt.
borrar_contrasena: Borrar contraseña almacenada en el sistema.
ayuda: Imprimir en consola lista de comandos disponibles.
acerca_de: Imprimir en consola parámetros del programa, curso y desarrolladores.

6. Lecciones Aprendidas

Este proyecto fue de provecho para comprender el funcionamiento y comportamiento de un árbol binario. Las rotaciones y distintas formas que puede tomar mientras se va modificando. Es una forma de abstraer el principio de una estructura de datos esencial como lo son los árboles. Otro aspecto de gran aprendizaje fue el manejo de archivos de cabecera en c. Como brindar permisos de acceso a funciones de manera privada y pública. Y por último fue de provecho aprendizaje sobre la teoría de encripción y su manejo en relación con el código.