Universidad del Rosario | 370 Años Formando proyectos de vida al servicio del país | Escuela de ingeniería, Ciencia y Tecnología

Manual técnico: Proyecto Algoritmos

Integrantes: Manuel Cruz Garrote y Mariana Sandoval Garzón

Enlace Repositorio:
https://github.com/ManuelCrzUR/Proyecto_Algoritmos_y_Estructura_de_Datos.git

1. Resumen Ejecutivo

El presente proyecto busca ser un apoyo educativo para nuevos estudiantes que estén cursando la materia de Estructura de datos. Se ha identificado como problemática en las aulas, que los estudiantes tienden a enfocarse más en la construcción y la implementación de las funciones de los conceptos vistos en clase. Dicha situación impide que, en la resolución de problemas, los estudiantes desarrollen su pensamiento analítico y creatividad. Por ello, se plantea hace un API que recoja todas las funciones y construcción de los temas Árboles Binarios y Mapas. De esta manera, este tipo de librería contendrá la construcción de este tipo de estructura de datos y las principales funciones que se utilizan en los problemas aplicados. Así, el estudiante tendrá disponible todas las herramientas que necesita para poder desarrollar las actividades propuestas en clase relacionadas con estos temas y reforzar las habilidades de programación que necesitan para su vida profesional.

2. Funcionalidad de la solución computacional

La solución computacional se compone de una librería por cada tema y un archivo ejecutable. Cada librería será un archivo de extensión "*.h*" donde estarán contenidas las clases y principales funciones que componen y se usan en los temas Árboles Binarios, Mapas y Tablas Hash. De este modo, el usuario podrá incluir las librerías en su archivo ejecutable y usar el contenido de estas a su disposición y necesidad. Además, se incluirán archivos de extensión "*.cpp*" que contendrán las funciones y clases en cada librería para que el usuario pueda estudiar el funcionamiento. Esta solución ayuda para el problema expuesto ya que permite facilidad y practicidad para el estudiante a la hora de la solución de problemas, de manera que tendrá las funciones hechas a la mano para implementarlo en los algoritmos propuestos para el problema.

3. Objetivos Específicos

- **Primer Objetivo***: Desarrollar en un API la implementación de los tipos de datos abstractos pilas, colas y mapas. Esto incluye la sintaxis de su declaración y las principales operaciones que se usan en problemas que involucren este tipo de datos.*

  Este objetivo se cumplió a su totalidad ya que se implementaron distintas librerías con clases y funciones principales de cada tema. Sin embargo, hubo un cambio parcial en el objetivo debido a que se cambio el tema de pilas y colas por árboles binarios por conveniencia del curso.

- **Segundo Objetivo:** *Implementación de una documentación clara y concisa en el código perteneciente al apartado de árboles binarios, la cual permita a las personas dar un punto de vista más claro y detallado acerca del funcionamiento del código presentado*

Este objetivo se cumplió a su totalidad y no hubo cambios respecto a este. En cada uno de los archivos de código se incluyeron comentarios claros que incluía por cada función una descripción, explicación de los parámetros y lo que retornaba. Además, se incluye un documento técnico y de usuario como guía.

4. Herramientas de programación

- Árboles binarios
- Mapas
- Tablas Hash
- Apuntadores
- Clases y funciones
- Librería map
- Librería list
- Librería String
- Repositorio Github
- Archivos de extensión .h

5. Descripción del programa

Nombre de los archivos:

Librerías

- mapas.h
- arbolesbinarios.h
- hash.h

Funciones de las librerías:

- mapas.cpp
- arbolesbinarios.cpp
- hash.cpp

Archivo que contiene el ejecutable

- main.cpp

Clases y Funciones mapas.cpp y mapas.h

- class Mapa
- Función: void Insertar
- Función: int ObtenerValor
- Función: bool ExisteClave
- Función: void EliminarElemento
- Función: bool EstaVacio
- Función: int ObtenerTamaño
- Función: void MostrarElemento

Clases y Funciones arbolesbinarios.h y arbolesbinarios.cpp

- class NodoAB
- Función: NodoAB *insertar
- Función: void PreOrden
- Función: void InOrden
- Función: void PosOrden
- Función: int buscar
- Función: int Sumar
- Función: int contar
- Función: NodoAB *encontrar_nodo
- Función: int camino
- Función: int peso
- Función: int hojas

Clases y Funciones hash.h y hash.cpp

- class HashMap
- Función: int funcionHash
- Función: void insertar
- Función: void remover
- Función: string obtenerValor
- Función: bool estaVacio
- Función: int obtenerTamaño
- Función: void mostrarMapaHash()

Flujo del programa (main.cpp):

1. Se incluyen las funciones del código *arbolesbinarios.h*, mediante la notación *#include "arbolesbinarios.h"*
2. Se incluyen las funciones del código *mapas.h* mediante la notación *#include "mapas.h"*
3. Se incluyen las funciones del código *hash.h* mediante la notación *#include "hash.h"*
4. Accede a las 7 funciones del código *mapas.h*
5. Accede a las 11 funciones del código contenido en *arbolesbinarios.h*
6. Accede a las 6 funciones del código contenido en *hash.h*

Tabla 1. **Descripción del funcionamiento del contenido de los archivos arbolesbinarios.cpp y arbolesbinarios.h**

Clases y Funciones	**Descripción Detallada**
class NodoAB	La clase NodoAB, inicializa el nodo o elemento del árbol binario y posee tres atributos: - valor: variable de tipo int, el cual representa el valor numerico de cada Nodo - sai: Es un apuntador a un NodoAB que representa al hijo izquierdo del árbol

	- sad: Es un apuntador a un NodoAB que representa al hijo derecho del árbol
Función: NodoAB *insertar	La función Insertar, como lo indica su nombre añade un nodoAB al árbol binario. La condición para añadir consiste en que los valores menores a la raíz van en el subárbol izquierdo y los otros en el derecho. Retorna un apuntador a un NodoAB que representa la raiz del árbol.
Función: void PreOrden	La funcion PreOrden recorre el arbol e imprime el valor de sus nodos empezando por la raiz, luego aplicando PreOrden en el subarbol izquierdo y luego en el subarbol derecho**.**
Función: void InOrden	La funcion InOrden recorre el arbol e imprime el valor de sus nodos empezando por aplicar InOrden en el subarbol izquierdo, luego la raiz y finalmente aplicar InOrden en el subarbol derecho.
Función: void PosOrden	La funcion PosOrden recorre el arbol e imprime el valor de sus nodos empezando luego aplicando PosOrden en el subarbol izquierdo, luego PosOrden en el subarbol derecho y finalmente la raiz
Función: int buscar	La funcion buscar, recibe un número y busca si ese valor corresponde a un NodoAB en el arbol. Retorna: 1: Si el valor ingresado corresponde a un NodoAB que esta en el árbol 0: Si el valor ingresado no corresponde a un NodoAB que esta en el árbol
Función: int Sumar	La funcion Sumar recibe la raiz de un arbol y suma todos los nodos que hay en dicho arbol. Retorna: 0: Si el arbol esta vacio El valor entero de la suma de los valores de todos los nodos del arbol.
Función: int contar	La función contar cuenta los nodos cuyos valores sean menores a el numero entero recibido como argumento de la funcion. Retorna: 0: Si el arbol esta vacio El número de nodos en el arbol cuyos valores sean mayores al número con el que se desea comparar.
Función: NodoAB *encontrar_nodo	La función encontrar_nodo recibe un número y encuentra el nodo del arbol cuyo valor corresponde al número recibido. Retorna: Un apuntador a un nodo de la clase NodoAB que corresponde al nodo del arbol cuyo valor corresponde a num
Función: int camino	La funcion camino recibe dos numeros y verifica si existe un camino para llegar del numero 1 al numero 2. Retorna: 0: Si no existe camino entre el nodo que corresponde a num 1 y el nodo que corresponde a num num2 1: Si existe camino entre el nodo que corresponde a num 1 y el nodo que corresponde a num num2
Función: int peso	La funcion peso cuenta la cantidad total de nodos que hay en el árbol. Retorna: 0: Si el arbol esta vacio El numero de la cantidad total de nodos en el arbol
Función: int hojas	La funcion peso cuenta la cantidad total de hojas que hay en el árbol. Retorna: 0: Si el arbol esta vacio

	El numero de la cantidad total de hojas en el arbol

Tabla 2. **Descripción del funcionamiento del contenido de los archivos mapas.cpp y mapas.h**

Clases y Funciones	**Descripción Detallada**
class Mapa	La clase Mapa almacena en un privado una estructura de mapa, la cual sirve para almacenar la información en este apartado. Gracias a lo anterior se puede ejecutar las funciones sin usar parámetros innecesarios y hacer el flujo del código mas rapido y eficiente.
Mapa()	Función que construye el objeto mapa.
Función: void Insertar	Función que inserta el par (clave, valor) en un mapa. Esta función recibe un string y un entero como parámetro, siendo el string la clave del valor entero. Esta función añade esta pareja al mapa respectivo
Función: int ObtenerValor	Función que devuelve un valor de tipo entero, que se encuentra en una clave del mapa. Esta función recibe un string que pertenece a una clave del mapa. Cuando encuentra la clave, accede al valor a esta y lo retorna.
Función: bool ExisteClave	Función que comprueba si una clave existe en un mapa. Esta función recibe un string que pertenece a la clave, comprueba en el mapa si la clave esta contenida en el mapa.
Función: void EliminarElemento	Función que elimina los elementos en una clave. Esta función recibe un string que pertenece a una clave dentro del mapa y elimina la información que se encuentre contenida en la clave.
Función: bool EstaVacio	Función que comprueba si un mapa está vacío. Esta función evalúa si el mapa tiene información almacenada en sí.
Función: int ObtenerTamaño	Función que identifica la cantidad de elementos en un mapa. Esta función identifica la cantidad de elementos que se encuentran en un árbol binario y las retorna en un entero
Función: void MostrarElemento	Función que imprime en la consola el contenido del mapa. Esta función recorre todos los elementos que existen y pertenecen al mapa, usa un ciclo for con un iterador el cual imprime en la consola el contenido de los mapas en el formato "llave - valor"

Tabla 3. **Descripción del funcionamiento del contenido de los archivos hash.cpp y hash.h**

Clases y Funciones	**Descripción Detallada**
class HashMap	La clase "HashMap" realiza la implementacioń para un mapa hash, en el privado mantiene una longitud de 10, guarda su estructura y almacena la función hash, la cual es responsable de dividir el almacenamiento del mapa.
Función: int funcionHash	Función que retorna el índice de la ubicación de la información en la tabla hash. Esta función recibe un entero el cual se procesa dependiendo de la función hash, en este caso lo divide y toma el valor entero.
Función: void insertar	Función que añade la pareja (Llave, Valor) en el espacio perteneciente en la tabla hash. Esta función, inserta información a la tabla hash, esta mantiene un formato de parejas ordenadas como

	antes se describio. Si el espacio de memoria se encuentra ocupado, este realiza un pushback, para que se almacene de ultimas en los elementos correspondientes a ese indice.
Función: void remover	Función que elimina una pareja ordenada de un espacio de memoria dentro del mapa hash. Esta función elimina la pareja ordenada (Llave, Valor) usando la Llave de un elemento
Función: string obtenerValor	Función que retorna la informacioń que se encuentre en una llave. Esta función obtiene el Valor de la pareja (Llave, Valor) accediendo al indice en el cual se encuetra almacenado.
-Función: int obtenerTamaño	Función que retorna la cantidad de elementos que tiene el mapa hash almacenados. Esta función recorre todos los indices y obtiene la longitud de cada uno y retorna la suma de todos
Función: bool estaVacio	Función que retorna si la llave ingresada contiene algun elemento. Esta función, retorna un booleano dependiendo si la existe un valor en la clave ingresada.
Función: void mostrarMapaHash()	Función la cual imprime el mapa hash. Esta función se encarga de recorrer todo el mapa hash e imprimirlo en la terminal. La impresión se divide entre los indices del mapa y las parejas ordenadas que se encuentren en cada uno de los indices pertenecientes al mapa hash.