**YZM 2124**

Veri Yapıları

**ÖDEV 2 RAPORU**

1. Giriş

Ödevin amacı ‘sehir\_plaka\_kodları.txt’ dosyasında bulunan şehir ve plaka kodlarını okuyarak iki boyutlu bir dizi elde etmek ve bu dizideki plaka kodlarından bir ağaç veri yapısı oluşturmaktır. Oluşan ağacın görüntüsünü, derinliğini, dengeli olup olmadığını ekranda gösterir. İki plaka kodu girilmesi istenir. Girilen plaka kodlarının şehir isimleri ve düzey farkları ekrana yazdırılır. Son olarak ağaç veri yapısını preorder, inorder, postorder olarak sıralar ve yazdırır.

Ödevin tamamı main.cpp adlı dosyada yer alır. Dosyanın içinde bulunan fonksiyonlar şunlardır:

void AgacEkle(Node\*& kok, const string& plaka) : Ağaca düğüm eklemek için kullanılır.

void AgaciYazdir(Node\* kok, const string& prefix = ‘’’’): Ağacın görüntüsünü ekrana yazdırmaya yarar. *prefix* parametresi, düğümleri ekrana yazdırırken kullanılan ön ek (prefix) stringini belirtmek için kullanılır. Bu parametre, düğümün seviyesini ve ağaçtaki konumunu temsil eden bir önek stringidir.

İnt AgacinDerinligi(Node\* kok): Ağacın derinliğini hesaplayıp döndüren fonksiyondur.

bool AgacDenge(Node\* kok): Ağacın dengeli olup olmadığını bulup true yada false döndürür.

Node\* AgacAra(Node\* kok, const string& plaka): Plaka kodunu ağaçta arama yapmayı sağlayan fonksiyondur.

int DuzeyFarki(Node\* kok, const string& plaka1, const string& plaka2): Girilen iki plaka kodu arasındaki düzey farkını bulmayı sağlayan fonksiyondur. AgacAra fonksiyonunu kullanarak plaka1 ve plaka2 değerlerine sahip düğümleri belirler. Eğer plaka kodu bulunamazsa -1 değeri döner. OrtakAta isimli bir düğüm oluşturulur. OrtakAta düğümü plaka1 ve plaka2 değerlerini büyüklük küçüklük durumuna göre döngüyü tarar. Ortak ata bulunamaz ise fonksiyon -1 değerini döner. İki düğümün düzeylerini hesaplamak için duzey1 ve duzey2 değişkenleri oluşturulur. Her iki değişkene 0 değeri atanır. Duzey1 değeri temp adında geçici bir düğüm kullanılarak plaka1 i bulana kadar sağa veya sola hareket ettirilir. Aynı şekilde duzey2 değeri plaka2 değerine sahip düğümü bulmak için hesaplanır. Son olarak duzey1 ve duzey2 değerleri arasındaki mutlak fark hesaplanır ve bu değer döndürülür.

void PreorderYazdir(Node\* kok): Ağacın preorder sıralayarak ekrana yazdırır.

void InorderYazdir(Node\* kok): Ağacın inorder sıralayarak ekrana yazdırır.

void PostorderYazdir(Node\* kok): Ağacın postorder sıralayarak ekrana yazdırır.

void AgaciTemizle(Node\* kok): İşlemler bittiğinde ağacı bellekten çıkartarak belleği boşaltan fonksiyondur.

int main() : İçinde MAX\_SEHIR\_SAYISI adında depolamak istenen maksimum şehir sayısını tutan integer bir değişken bulunmaktadır. Dosya okuma işlemi bu fonksiyon içerisinde gerçekleşmektedir. Sehirler adında iki boyutlu dizinin oluşturulduğu ve dosyada okunup verilerin ayırt edilerek diziye kaydedildiği fonksiyondur. Yukarıda belirtilen fonksiyonlar burada çağrılır. Kullanıcıdan plaka kodu bu fonksiyon içerisinde istenip bu fonksiyon içerisinde değere karşılık gelen şehir bulunur.

1. Geliştirme ve Çalıştırma Ortamları

Ödev Visual Studio kullanarak geliştirildi

1. Veri Yapıları ve Değişkenler

Ödevi geliştirilirken birden fazla veri yapısı kullanıldı. Dosya okuma işleminden elde edilen veriler diziler ile tutuldu. Daha sonra bu tutulan verilerden ağaç veri yapısı oluşturuldu.

1. Program Akışı

İlk olarak ‘sehir\_plaka\_kodları.txt’ dosyası açılır. Dosyada bulunan veriler okunur. Ardından plaka kodları ve şehir isimlerinden iki boyutlu bir dizi oluşturulur ve okunan veriler kaydedilir. Plaka kodlarından oluşan bir ağaç veri yapısı oluşturulur. Ağacın derinliği ve dengeli olup olmadığı bulunarak yazdırılır. Daha sonra ağacın görüntüsü ekrana bastırılır. Kullanıcıdan iki adet plaka kodu girmesi istenir. Girilen plaka kodlarından plakanın hangi şehre ait olduğunu ve iki plaka arasında bulunan düzey farkını bulur. Son olarak ağaç veri yapısını preorder, inorder ve postorder şeklinde sıralayarak ekrana yazdırır.

1. Sonuç

Ağaç veri yapısını daha iyi anlama ve deneyimleme imkanım oldu. Ağaç veri yapısı üzerinde arama işlemlerinin nasıl gerçekleştiğini daha iyi anladım ve ağaç veri yapıları için sıralamaları daha iyi öğrendim. C++ da bulunan dosya işlemleri hakkında daha fazla bilgi sahibi oldum ve kendimi geliştirdim. Derste gördüğüm teorik bilgileri pratiğe çevirdim. Visual Studio ya olan hakimiyetim arttı ve kullanım hızı kazandım.