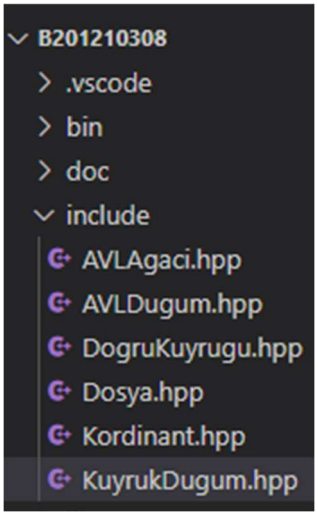
Ad : Fırat

Soyad : Turan

Öğrenci Numarası : B201210308

Grup : 2.Öğretim B gurubu

E-Mail : firat.turan2@ogr.sakarya.edu.tr



Veri Yapıları Ödev-2

Yapmış olduğum projede program, açıldığı gibi Noktalar.txt dosyasını okuyacak ve her satırda bulun sayıları x,y ve z kordinatı şeklinde bir kuyruğa ekleyecek kuyruğa ekleme yaparken orjine yakın olan kordinat öncelikli olacak. Kuyruğa eklenen tüm kordinantlar txt dosyasındaki sıraya göre aralarındaki mesafe toplanacak ve her satırdaki toplam değer ile bir AVL ağacı oluşturulacak. Yazdırma işlemi postorder şeklinde AVL ağacındaki kuyrukların içinde bulunan koordinatların orjine uzaklığına göre yazdırılacak.

Sol taraftaki görselde görüldüğü gibi 6 farklı başlık oluşturdum.

Dosya

```
class Dosya
{
public:
    void dosyaOku(string dosya);
    void kordinantBul(string line);
    AVLAgaci agac;
};
#endif
```

Noktalar.txt dosyasındaki verileri satır bazında okuma için kullandık.

Okuduğumuz satırdaki x,y ve z kordinatlarını çektik

Okunan değerleri ağaca aktarmak için tanımladık

AVLDugum

```
class AVLDugum
{
public:
    AVLDugum(DogruKuyrugu* kuyruk);
    int veri;
    DogruKuyrugu* kuyruk;
    AVLDugum* sol;
    AVLDugum* sag;
};
```

DogruKuyrugu cinsinden tutan bir düğüm tanımladık

Kuyruğun toplam uzunluğunu tutan bir değişken tanımladık

AVL ağacı için sol ve sag değişkenlerini tanımladık

AVLAgaci

```
class AVLAgaci
{
public:

    AVLAgaci();
    ~AVLAgaci();
    void ekle(DogruKuyrugu* kuyruk);
    int yukseklik();
    void postOrder();
private:
    AVLDugum* solaDondur(AVLDugum* AVLDugum);
    AVLDugum* sagaDondur(AVLDugum* AVLDugum);
    int yukseklik(AVLDugum* aktifAVLDugum);
    AVLDugum* ekle(DogruKuyrugu* kuyruk,AVLDugum* aktifAVLDugum);
    void postOrder(AVLDugum* aktif);
    AVLDugum* kok;
};
#endif
```

Ağacımıza ekleme yapmak için ekle metodu tanımladık

AVL ağacı kontrolleri için yukseklik metodu tanımladık

PostOrder okuma için postOrder metodu tanımladık

Sola ve Sağa döndürme işlemleri için tanımladık

KuyrukDugum

```
KuyrukDugum::KuyrukDugum(Kordinant* kordinant)
{
    this->kordinant = kordinant;
    sonraki=0;
}
```

Kuyruk işlemleri için Kordinant değeri tutan bir düğüm tasarladık

DogruKuyrugu

```
class DogruKuyrugu
{
public:
    DogruKuyrugu();
    void ekle(Kordinant* kordinant);
    void Listele();
    Kordinant* sonKordinant;
    int toplam;
private:
    int uzunlukHesapla(Kordinant* kordinant1,Kordinant* kordinant2);
    KuyrukDugum* ilk;
    KuyrukDugum* son;
```

Kuyruğa kordinant eklemek için ekleme metodu oluşturduk ekleme yaparken öncelik sıramasını ayarladık.

Listeleme metodu ile kuyrukta bulunan kordinatların orjine uzaklıklarını listeledik.

sonKordinant'ı toplam uzunluğu bulmak için kullandık.

Toplam uzunluğu tuttuk

2 kordinant arasındaki mesafeyi ölçmek için oluşturduk

Kuyruk için ilk ve son değerleri tanımladık.

Kordinant

```
class Kordinant
{
public:
    Kordinant(int x,int y,int z);
    int x,y,z;
    int uzaklikHesapla();
};

#endif
```

x,y ve z kordinatlarını tutmak için tanımladık.

Kordinatın orjine olan uzaklığını hesaplamak için tanımladık