Uygulamaya başlarken 2 farklı durum ile ilgilenmem gerektiği için iki parçaya bölüp o şekilde tamamladım. Birinci bölümde bağlıliste tasarım kısmını kurgularken, ikinci bölüm de ağaç çizdirme ve ağaç ile ilgili işlemlerini (aynalama, verihesaplama..) kurguladım.
BağlıListeTasarımı:
Bağlı liste tasarımında ekranda çıkacak düzeni oluşturabilmek için bilmem gereken en önemli şey bir alt satıra geçtikten sonra üst satıra geri dönemeyeceğimdi, bu sebeple tasarımını yaparken her bir satırı o şekilde işledim önce bir satırı yapıp daha sonra alt satıra indim. Tasarım düzeni (bir düğümü esas alırsak) Noktalar/ Bağlıliste de tutulan düğümün adresleri/Noktalar/ Bağlılistede tutulan düğümün verileri/ Noktalar/ Bağlıliste de tutulan düğümden sonraki düğümün adresleri/Noktalar. Bunu maksimum 10 düğüm yan yana gelecek şekilde tekrarlamam isteniyordu bende aynı sırayla giderek Noktalar/Hangi sıradaysam o sıranın indeksini dugumgetir fonksiyonu ile alıp o düğümün adresi/Noktalar/Düğümgetirle yine o düğümün verisi/Noktalar/ Düğümgetir fonksiyonu ile ilgili düğüme erişip onun sonraki düğümünün adresi/Noktalar şeklinde oluşturdum. Bizden istenen şey ekranda maksimum 10 adet Bağlı liste düğümünün gözükeceğiydi bunu sağlamak için başlangıç ve bitiş değerlerini parametre olarak aldım. Bir grup değişkeni oluşturup hangi grupta olduğumu kontrol ettim, gruplar en fazla 10 veri tutabiliyordu eğer indeksim 9u geçerse diğer gruba geçeceği için indeks 0lanıp grup değişkenim 1 artacaktı. Eğer indeksim 0'dan küçük olursa 1.grup haricinde 9 olarak değişecekti, 1.grupta ise 0'dan aşağı düşmeyecekti.Grupdaki eleman sayısı 10dan az ise bitiş düğüm sayısına eşitlenecekti dolaysıyla sınırları aşmayacaktım. Bu şekilde hataların önüne geçmeyi hedefledim. Ayrıca istenen verilere ulaşmak için bana baglilistedeki hangi verinin gösterileceği bir değişken olarak baglilisteindeksi lazımdı, bu baglilisteindeksini kullanarak düğümgetir fonksiyonuyla o düğüm ile ilgili işlemleri yapmama olanak tanıdı. W ye basıyorsam aynalama S ye basıyorsam sil işlemlerini yapmak için baglilisteindeks değişkenini, D ye basınca sağa A ya basınca sola gitmek içinde indeks değişkenini kullandım. Bağlıliste tasarımını bu şekilde oluşturdum.
Ağaç Oluşturma:
Ağaçların düğümlerinde istenen agaclar.txt dosyasındaki her bir satırın bir ağacı temsil etmesiydi bu sebeple bende her bir satırı dosyadan okuma ile agaclar.txt'yi okuyup her bir satırı alıp her satır için ağaç oluşturdum, satırdaki her bir elemanı ağaca char olarak ekledim. yani ağaç düğümlerinde char veri tipi tutuyordu ve bunun sağlanması için satırlarıağacyap adında bir fonksiyon tasarladım .txt dosyasını okuduktan sonra her bir satırı tek tek satırlarıagacyap fonksiyonu ile agaç yaptım. Dosyadan satırları okuyup ağaç oluşturma fonksiyonumu bu şekilde tasarladım.

Ağaç Çizdirme:
Ağaç çizdirme işleminde çizdirme yapmak için dikkat edilmesi gereken en önemli husus alt satıra geçtiğimde üst satıra tekrar erişemeyeceğim olduğu için bana verileri sırayla yukardan aşağı çizdirecek bir ağaç yazdırma işlemi lazımdı. Bunun için en iyi aday levelorder oldu levelorder ile verileri yukardan aşağıya doğru sırayla yazdırabilecektim. Verileri dolaşmak eklemesini yapmak için kuyruk veri yapısından faydalandım. Hazır veri yapısı kullanmak yerine kendi kuyruk sınıfımı oluşturdum. Ağacı çizdirme yaparken 3 farklı kuyruk ile bunu yapmam gerektiğini düşündüm. 1.kuyruk verileri yazmak için, 2.kuyruk verilerin hemen üstünde bulunacak gösterici (dalın ucu) için, 3.kuyruk ise sol-ebeveyn, ya da ebeveyn-sağ, ya da sol-ebeveyn-sağ durumlarında çocukları ile kendi arasında bulunan noktaları göstermek içindi. Tasarımı yaparken önce elime kağıt kalem alıp bu durumun nasıl oluşacağını hesaplamaya çalıştım, sonra fark ettim ki belirli bir örüntü oluşturabileceğimdi oda şöyleydi örneğin seviyesi 4 olan tüm verileri dolu bir ağaç düşünelim. Kağıt kalemle çizince en aşağıda kalan yapraklar arasındaki mesafeyi 1 kabul edersek bir üst yapraklar arasındaki mesafe 3 oluyor bir üste yapraklar arasındaki mesafe 7 bu örüntü bu şekilde katlanıyor 1-3-7-15.. yani

(2^n -1) formülünü yakaladım. Tabi aşağıdan yukarıya doğru bu şekilde olduğu için bir seviye değişkeniyle bunu yukardan aşağıya azalan şekilde yakalamak için n değerini (yuksek-seviye) şeklinde tanımlamam gerekti bu sayede istediğim boşluk örüntüsünü yakalamış oldum, daha sonra verileri yatay olarak yazdırmam gerektiği için(bir üst satıra geri dönememe sorunsalı) kuyruktan faydalandım. Bunun içinde bir kat değişkenine ihtiyacım vardı. Kat değişkeni sayesinde verileri kuyruğun içine alıp veri yazdırma yada nokta bırakma işlemini yapıp ilerleyip katı bitirene (0layana kadar ilerliyoruz) katın bittiğini tanımlamak içinde her bir seviyedeki verileri incelediğimde: 0.seviyede 1, 1.seviyede 2,2.seviyede 4 buradan yakaladığım örüntü ise (2^seviye=kat ) eşit olma durumu oluyor eğer bu durum sağlanırsa katı sıfırlayıp seviyeyi arttırıyoruz ki bi alt satıra geçelim. Aradaki tamamlayıcı noktalar içinde şöyle bir yol izledim. Durumları incelediğimde çizdirme yapabilmek için ya solu vardır, ya hem solu hem sağı vardır, ya sadece sağı vardır yada solu ve sağı yoktur. Bu bilgiler ışığında gerekli şart yapısını kurdum. Dikkat etmemiz gereken husus birbirlerine girmemesi için solu var ise içinde sağıda var ise durumu,kısaca özeti(-if(sol) {solunİslemi if(sag){soldanSonrakiSaginİslemi} }else if(sag){sadeceSaginİslemi}-) eğer düğümün solu var ise solundan itibaren (veri-noktabirak-veri) solu ve sağ var ise (veri-noktabirak-veri-noktabirak-veri) sadece sağı var ise (veri-noktabirak-veri) geri ye boş kalacak noktalar içinde (bosluk-boslukbirak-bosluk-boslukbirak-bosluk) seklinde yapınca düzen tam olarak sağlanmış oluyor. Ağaç çizdirme işlemini bu şekilde tamamladım.

Ağaç Verisi Hesaplama:

Ağacın verisini hesaplamak için verilen kural karakterlerin ASCII değerlerine göre sol düğümlerin verisi 2 ile çarpılacak sağ düğümlerin verisi olduğu gibi yazılacak bunu sağlamak için parametre olarak çarpan değerini aldım, eğer düğüm sol ise 2 ile çarpılacak sağ ise 1 ile çarpılacak şekilde kurguladım veri hesapla adında kurduğum fonksiyonda bir toplam değişkeni atamasını yaptım sol ve sağ veri bitene kadar fonksiyon kendi kendini çağıracak her sol için çağırdığında çarpan katsayım 2 olacak her sağ için çağırdığında çarpan katsayım 1 olacak bunları da sürekli toplam değişkenime ekleyecek burada dikkat etmem gereken husus verileri char veri tipinde tuttuğum için tek tek integer veri tipine eklenme yapılması yani başka bir deyişle eğer c++ da böyle bir ifadeyi ekrana yazdırırsam int sayi = ‘A’ bu bana 65 yani A nın ASCII değerini döndüreceği için toplamı int olarak aldığım zaman istediğim sonuca ulaşacağımdı en son toplamı return edince verihesaplama fonksiyonum tamamlanmış oldu.

Ağaç Aynalama:

Ağaç aynalama işleminde bizden istenen çizdirilen agacin ayna görüntüsünün oluşması ve bu doğrultuda verisinin de değişmesiydi. Bunun için ağacın düğümlerindeki sol ve sağ düğümlerin yerini değiştirmem gerekiyordu, bunu sağlamak için geçici bir değişken oluşturdum daha sonra bu değişkenle tüm düğümler için sağ ve sol düğümlerin yerini değiştirdim fonksiyonu tekrar kendisi için çağırınca tüm düğümlere erişerek eğer varsa sağını sola varsa solunu sağa döndürecek fonksiyonu yazdım.

Ağacı BağlıListe düğümlerinde tutma:

Ağacın düğüm sınıfı char veri tipi tutuyor, Bağlılistenin düğüm sınıfı ağacın işaretçilerini tutuyor,ağacı bağlıliste düğümlerinde tuttuğumuzdan dolayı ilgili düğüm silindiğinde bellekte çöp oluşmaması için ağacında silinmesi gerekiyor, bu yüzden ağacın yokedicisi için bir sil fonksiyonu tanımlayıp ilgili tüm düğümleri silmesini sağladım. Bu sayede bir düğüm silindiğinde düğümdeki ağaçta siliniyor. Bağlantıyı sağladığımız için bağlılisteden ağaca ağaçtanda işlemlere erişim sağlayabildim. Ağaç ve bağlıliste bağlantısını bu şekilde tasarladım

Ad:Furkan
Soyad:KIRAK
Öğrenci Numarası:B241210300