Doğan Sağlam
2004510032

2. **Ödevin Raporu:**

Program Ne Yapıyor:

Bu program Linux işletim sistemindeki “info“ adlı dökümanların gezilmesini kolaylaştıran programın küçük bir benzeridir.Başlangıçta “dir.txt “ adlı dosyaya erişir ve bu dosyada belirtilen menü elemanlarına veya metin içindeki linklere gidebilmesini sağlar.Kısaca kullanıcının istediği konuyla ilgili bilgiye kullanıcıyı doküman sistemi içinde dolaştıran bir program sayesinde ulaştırır. Kullanıcını yapacağı tek şey bir “link” olduğu belirtilen konuyla ilgili bilgiye ulaşmak için gerekli komutları kullanmaktır.

Benim yaptığım test programı “dir.txt” dosyasına ek olarak “G_S.txt” ve “E_I.txt” adlı iki dosya daha vardır.Tüm bu dosyaların içindeki bilgileri “info.info” adlı “gnu” tabanlı “info“ programlarında da kullanılan dosyadan kopyaladım ve bunları kısalttım.Bazı bilgileri ise kendi programıma göre değiştirdim.

Her konunun alt konuları olabilir ve bir dosyada sınırsız sayıda konu (“dir.txt” adlı dosya hariç :Bu dosyada yalnızca bir tane konu vardır.)olabilir.Bir konunun alt konusu başka bir dosyadaki herhangi bir konu olabilir.Konuları birer düğüm gibi düşünecek olursak ,birbirine bağlı ve hiyerarşik bir düzen ,yapmak istediğimiz programda ne tür bir veri yapı kullanabileceğimiz hakkında bize fikir verebilir.

Girdi dosyalarının biçimi:

Her bir dosya “[“karakteri ile başlar.Bu karakterden sonra konuyla ilgili düğümün ismi yazılır sonra bir iki nokta gelir ve düğümün kullanıcıya yansıtılacak ismi yazılır.(“Kullanıcı düğüm isimlerini çıktıda asla göremeyecektir”).Daha sonra yine bir “]” karakteri gelir ve satır karakteri eklenir.Bu satırda başka Bir şey yazılmamalıdır yazılsa da göz ardı edilecektir.

Daha sonraki satırlarda konuyla ilgili metin olabilir .Bu metnin içinde de “(” karakteri ile başlayan ve ardından yine “[” karakteri ile devam eden “linkler” vardır.Bu linkler başka dosyalara da gidebileceği için böyle bir durumda iki nokta işaretine dek düğümün yer aldığı dosyanın ismi yazılmalıdır.Eğer aynı dosyada bulunuyorsa buraya hiçbirşey yazılmaz.İkinci bir iki nokta işaretinden sonra düğümün programda kullanılacak ismi yazılmalıdır ve daha sonra bir “]” karakteriyle bitirilir.Fakat düğümün kullanıcıya yansıtılacakismi için “)” işaretine dek yer kalmıştır.Bu da yazıldıktan sonra “)” işaretiyle “link” kapanır.Metnin devamı gelebilir veya menüde bulunan “linklere” geçiş yapılabilir. Menüde bulunan linklere geçiş bir satırda yalnız başına duran “* Menu:” karakter katarından anlaşılır.Bu satırdan sonra bir satır atlanarak her bir “*” işaretinden sonra aynı metin içindeki “linklerin” biçiminde olacak menü linklerini belirten biçim gelir.Bu link biçiminin yazılmasından sonra her satıra o linkin açıklaması olan bir miktar metin gelebilir.Her bir menü linki yalnız bir satıra sığmalıdır.Şunu da belirtmekte önem var sayfanın geneli için her bir satır en fazla 80 karakterden oluşmalıdır.Her bir düğüm “%%” işaretinde biter.

Arayüz Dosyaları:

Programın tüm kaynak kod dosyaları “hw2” adlı ad uzayının içinde yer alır ve bu ad uzayını kullanır.

info_tree:

Programın ana yapısı olan “info_tree” adlı sınıfı da buna göre hazırladım.C++ ' ta “string” sınıfı standart kütüphane içinde yer aldığından ve bu sınıfın nesnelerine çok fazla ihtiyaç duyduğumdan dosyanın başında “string” sınıfını dosyaya dahil ettim.Ayrıca özyinelemeli bir ağaç yapısını gerçekleştireceğim ve her bir ağacın en az sıfır en fazla “n” tane daha alt ağacı olabileceğini düşündüğüm için her bir ağaç nesnesinin bir “vektör” alt alanı olmasında karar kıldım.Böylece alt ağaçlar arasında ileri geri gitmenin yanısıra bu işi dinamik ve hızlı bir şekilede yapabileceğimi düşündüm.Ayrıca her bir alt alanın “next” ve “previous” yerine bir tane “up” alt alanı olmasını sağlayabileceğimi düşündüm.Bu “up” bir üst seviyedeki düğümü gösterecek ve her bir düğüm

içinde bulunduğu seviyeyi bu göstericinin metotları sayesinde öğrenebilecek.Ayrıca önemli bir nokta daha yapımı gerçekleştirebilmek için gerekli alt alanları koyabileceğim herkesin rastgele erişemeyeceği "private" erişim belirleyicili bir iç sınıf ("info_node") gerçekleştirmeye ihtiyaç duydum.Böylece saklı alt alanlarımla ilgili, istediğim bilgileri kullanıcıya açık "public" ve "info_tree " nesnesine yöneltilen fonksiyonlarla elde edilebilir ve değiştirilebilir kıldım.
Bu fonksiyonların tümünün gerçekleştirimleri "info_tree.cxx" adlı dosyada yaptım.Ayrıca dikkat edileceği üzere fonksiyonlara aktarılan argümanları genellikle gösterici türünden tanımladım.Aslında burada hızdan bir miktar kazanmayı düşünürken kullanıcının orijinal verinin bütünlüğüne de zarar verebileceğini düşündüm fakat kopyalama yapıcı metodunda çözemediğim bir hata olmasından dolayı ve bu programın öğrenmek amacı gütmesi sebebiyle bu konu üzerinde fazla duramadım. Şimi gerçekleştirime bakalım..
info_tree.cxx:
Bu dosyada kullanılış biçimleri "info_tree" dosyasında belirtilmiş olan "info_tree" sınıfına ait fonksiyonların gerçekleştirimini verdim.Bunun için önce " info_tree" dosyasını bu dosyanın içeriğine dahil ettim.Foksiyonların gerçekleştirimine bakacak olursak:

info_tree(int addr,string* n_n,string* d_n,int type):
Bu yapıcı metod dosyadan okunan bir düğümün o anda hangi özelliklerinin anlaşılabileceği düşünülerek yazıldı.Düğümün yalnızca adresi,düğüm adı ve düğümün dosyasının adı ilk anda bilinebileceği için yapıcı fonksiyona bu bilgiler sırasıyla gönderiliyor.

~info_tree(void)
Yıkıcı metod içinde vektörüm içi boşaltılıyor.

info_tree(const info_tree& item)
Burada referans olarak geçirilen değerin alt alanlarnın değerleri mesajı alanınkilere kopyalanıyor.

Get_Subtree(string* dis_n,int type)
Geçirilen isme("kullanıcıya yansıtlır isim") ve tipe göre("ilgili düğüm menü ya da metin linki mi?") arama yapılır ve bir gösterici dödürülür.Eğer aranan bulunamadıysa "null" değerinde bir gösterici döner.

info_tree* info_tree::Get_next()
Mesajı alan nesnenin üst seviye nesnesinin vektör alt alanına bakar.Bu vektörün içinde mesajı alan nesnenin indeksinden bir sonraki nesnenin göstericsini döndürür.Burada vektörün"at" adlı metodunu kullandım.

void info_tree::Set_Subtree(info_tree* item)
Mesajı alan nesneyenin vektör alt alanına ,geçirilen argümen geliş sırasına göre eklenir.Burada vektörün "pushback" metodunu kullandım.

```
string info_tree::Get_disname()
  {
            return root->dis_n;
  }
```

Yukarıda görüldüğü gibi "Get" ile başlayan tüm metodlar "root" adlı "info_node" sınıfına ait nesnenin alt alanlarından birini döndürürler.

İnforeader:
Bir sınıf değildir sadece aynı isim uızayındaki fonksiyonlardan oluşan ve “info_tree” sınıfını kullanan bir modüldür.

Inforeader.cxx:
Bu dosyada metin dosyalarının okunması bunların gerekli yapıya(“info_tree”) yerleştirilmesi ve çıktı ortamına yazdırılması için gerekli fonksiyonlar vardır.

void Adjust_Nodes(string* link,info_tree* up,int type)

Bu fonksiyon “readinfo” adlı fonksiyon tarafından düğümü oluşturması için çağırılır ve daha sonra yine bu fonksiyonu çağırarak bir düğümden gidilebilecek son düğüme kadar gidip bunu yapıya eklemeyi sağlar.Burada geçirilen “up “ argümanına yeni oluşturulan düğüm eklenerek arada ikili bağ kurulur.”link” adlı “string” argümanın çözülmesi sayesinde üretilir yeni düğüm.Geçirilen tip ise bu düğümün bir menüde mi olduğunu belirtir.

info_tree* readfile(string* filename,info_tree* node_n)
Geçirilen ddosya ismindeki dosyayı “input file stream” nesnesi ile açar ve satır satır ilerlerken dosyadaki düğümleri çözümler bu arada her bir düğümün başlangıç adresini belirler ve Adjust_Nodes adlı metodu çağırarak öz yinelemeyi sağlar.

char disp(info_tree* dis)

Başlangıçta “windows “işletim dizgesinin system fonksiyonu olan 'cls' ile ekranı temizler.Geçirilen düğümle ilgili bilgileri basar ve düğümün olduğun dosyayı açarak düğümün başladığı yerden itibaren her bir “link”in yualnızca kullanıcıya yansıtılacak bölümlerini yazarak ilerler.Sonunda dosyayı kapatır ve kullanıcıdan bir seçenek ister.

test_info.cxx:

Burada“readinfo” dosyasının döndürdüğü gösterici yazdırıcı fonksiyona(“disp”) gönderilir.Daha sonra bir döngüde seçeneğe göre işlemler sürerek kullanıcı çıkış isteyene kadar (seçenek istenildiğinde “q” girerek) program devam ediyor.