* **BAĞLI LİSTE UYGULAMASI**

**Hakan Keleş**

**Bilgisayar Mühendisliği Bölümü**

**Kocaeli Üniversitesi**

**180201036**

**hakankeles4152@gmail.com**

Özet

Coğrafi Bilgi Sistemi (CBS), dünya üzerindeki karmaşık sosyal, ekonomik, çevresel vb. sorunların çözümüne yönelik mekâna/konuma dayalı karar verme süreçlerinde kullanıcılara yardımcı olmak üzere, büyük hacimli coğrafi verilerin; toplanması, depolanması, işlenmesi, yönetimi, mekânsal analizi, sorgulaması ve sunulması fonksiyonlarını yerine getiren donanım, yazılım, personel, coğrafi veri ve yöntem bütünüdür. Bu proje kapsamında, şehirlerin komşuluk ilişkilerini dikkate alan çok basit bir harita verisi üzerinde bağlı listelerle ilgili temel işlevleri yerine getiren bir uygulama yapılması beklenmektedir.

# Giriş

Bağlı liste uygulaması konsoldan kullanıcı için oluşturulan bir menü aracılığı ile uygulanan bir uygulamadır.

# Temel Bilgiler

Proje geliştiriminde tümleşik geliştirme ortamı olarak. ”NetBeans” in 8.2 sürümü kullanılmıştır.

# Yazılım Tasarımı

Proje geliştirilirken yapı sehirler.txt dosyasını dinamik oluşturulan city struct yapısına formatlı okuma yapılıp aktaracak şekilde tasarlanmıştır.

3.1.1 Fonksiyonlar

void add\_end ():

Parametre olarak struct city \*ct alan bu fonksiyon gelen her bir şehir elemanını oluşturulan bağlı listenin sonuna ekler.

void add\_head ():

Parametre olarak struct city \*ct alan bu fonksiyon gelen her bir şehir elemanını oluşturulan bağlı listenin başına ekler.

void add\_end2 ():

Parametre olarak struct neigbour \*neig alan bu fonksiyon gelen her bir komşu elemanını oluşturulan bağlı listenin sonuna ekler.

void add\_head2 ():

Parametre olarak struct neigbour \*neig alan bu fonksiyon gelen her bir komşu elemanını oluşturulan bağlı listenin başına ekler.

int search():

Parametre olarak herhangi bir şehrin plaka kadounu alan bu fonksiyon bağlı listede sırası ile dolaşarak plakanın listede olup olmadığını kontrol eder.

void add\_sequential ():

Parametre olarak struct city \*ct alan bu fonksiyon gelen her bir şehir elemanını daha önce BubleLink fonksiyonu ile sıralanmış olan listeye sıralanmış olarak ekler.

void list():

Parametre olarak FILE \*output alan bu fonksiyon başlangıçta struct city \*list tipinde bir parametre oluşturur ve kontrol mekanizması içerisinde her bir şehri tek tek dolaşarak şehrin özelliklerini FILE \* output dosyasına yazar.

void add\_sequential2():

Parametre olarak struct neigbour \*neig alan bu fonksiyon gelen her bir komşuyu sıralı ile oluşturulan şehir düğüm yapısına aktarır.

void list2():

Parametre olarak struct neigbour \*neig alan bu fonksiyon başlangıçta struct neigbour \*list tipinde bir parametre oluşturur ve kontrol mekanizması içerisinde her bir şehri tek tek dolaşarak şehrin özelliklerini çıktı olarak döndürür.

void delete\_city():

Parametre olarak struct city \*ct ve silinecek şehrin plaka kodunu alan bu fonksiyon başlangıçta struct city \*dell,\*pre şeklinde iki pointer tanımlanır ve gerekli işlemlerle şehir listeden silinmiş olur .

void list\_region():

Parametre olarak char region[] ve struct city \*ct alan bu fonksiyon başlangıçta struct city \*list şeklinde bir işaretçi oluşturur ve bu işaretçi listenin başını göstermektedir. Ardından her bir şehrin bölgesi ile gelen region[] parametresi karşılaştırılır ve uygun olan şehirler çıktı olarak yazılır.

void list\_number\_of\_neig():

Parametre olarak int number\_of\_ neigbour ve struct city \*ct alan bu fonksiyon başlangıçta struct city \*list şeklinde bir işaretçi oluşturur ve bu işaretçi listenin başını göstermektedir. Ardından her bir şehrin sahip olduğu komşu sayısı ile gelen number\_of\_neigbour parametresi karşılaştırılır ve uygun olan şehirler çıktı olarak yazılır.

void buble\_link():

Parametre almayan bu fonskiyon başlangıçta struct city \*ptr\_fun olan bir işaretçi tutmaktadır. Bu işaretçi liste üzerinde dolaşmakta ve listedeki her elemanı Buble Sort algoritmasına göre sıralamaktadır.

3.2 Karşılaşılan problemler ve çözüm yaklaşımları

3.2.1 Formatlı Okuyamama Hatası

3.2.1 Çözüm: fscanf ile belirlenen formatta okuyamadım ardından her bir satırı buffer[] içerisine attım ve virgüle göre ayrım yapıldıktan sonra şehir bağlı listesine aktarım yapıldı .

3.2.2 Şehir Silme Hatası

3.2.2 Çözüm:

3.2.3 Komşuların Plakalarının Sırası ile Aktarılamamsı Sorunu

3.2.3 Çözüm:

3.2.4 Aranan Şehrin Bulunamaması Sorunu

3.2.4 Çözüm:Struct yapısı üzerinde eleman eleman arama yapılmaktaydı ve istenen sonuç alınamaktaydı. Arama bağlı liste üzerinde yapıldı ve listenin ileriki şehir elemanı yine bir şehir olacağından doğru sonuç alınmış oldu.

3.4 Uygulamanın kullanımının anlatımı

Basit liste uygulaması konsoldan kontrol edilen bir

uygulamadır.

\* Program sehirler.txt içerisinde bulunan yapıyı şehir ve komşular yapılarına uygun şekilde aktarır.

\*Ardından kullanıcı için bir menü oluşturulmuştur.

\* Menüde kullanıcı belli başlı işlemleri seçerek ilerleme sağlanır.

\* Fonksiyonlara gönderilen parametrelerle output.txt dosyasına şehir isim ve özelliklerinin yazılması sağlanmaktadır.

# Kod Akışı

Bu bölümde kodun genel işlemleri adım adım anlatılacaktır.

• Uygulama kullanıcı tarafından çalıştırıldığı anda ilk olarak main ana fonksiyonu çalışacaktır . Ardından program dosya dizinine ulaşacaktır ve gerekli txt dosyasını çekip işlem yapmaya başlayacaktır.

•**Dosyanın çekilmesi işlemi;**

FILE\* infile;

FILE\* outfile;

infile = fopen(filename,"r");

outfile = fopen("output.txt","w");

• if(!dir){

printf("Directory was not found!\n");

exit(1);

}

char filename[12];

while( (sd=readdir(dir)) != NULL){

char \*ptr;

ptr = strstr(sd -> d\_name , search1);

if(ptr != NULL){

strcpy(filename,sd->d\_name);

}

}

•Ardından fgets fonksiyonu ile her bir satır tek tek alınarak char tipinde buffer değişkenine atılmaktadır.

fscanf(infile,"%[^\n]\n",buffer[k]);

•Buffer değişkeninde okunan değerler sırası ile struct city tipinde oluşturulan şehir sayısı kadar kapasitesi olan ct[] dizisine gönderilmektedir.

for(int i=0;i<number\_of\_city;i++){

ct[i].plate = atoi(buffer[i]);

strcpy(ct[i].cityName,cityName[i]);

strcpy(ct[i].region,region[i]);

add\_sequential(&ct[i]);

}

•Bu kısımlar tamamlandıktan sonra Menü kısmına gelinmektedir. Kullanıcıdan aldığı değerlere göre:

Case1: add\_sequential(&ct[number\_of\_city-1]);

ct = realloc(ct,number\_of\_city\*sizeof(struct city));

neig = realloc(neig,number\_of\_city\*sizeof(struct neigbour));

const char \*token2 = strtok(buffer2,",");

int line2 = 0;

while(token2 != NULL){

if(line2 == 1){

strcpy(ct[number\_of\_city1].cityName,token2);}else if(line2 == 2){

strcpy(ct[number\_of\_city-1].region,token2);

}

line2 ++;

token2 = strtok(NULL,",");

}break;

Case2:

printf("Please enter city info to delete it: ");

scanf("%s",buffer4);

int result\_of5 = search(atoi(buffer4));

if(result\_of5 != -1){

delete\_city(ct,result\_of5);

}break;

Case3:

printf("Please enter city info to search it: ");

scanf("%s",buffer5);

int result\_of6 = search(atoi(buffer5));

add\_sequential(&ct[number\_of\_city-1]);

break;

Case4:

list(outfile);

while(list != NULL){

fprintf(outfile,"%d %15s %3s %d ",list->plate,list->cityName,

list->region,list->number\_of\_neigbour);

for(int i=0;i<list->number\_of\_neigbour;i++){

fprintf(outfile,"->%d",list2->plate);

list2 = list2->next;

}

fprintf(outfile,"\n");

fprintf(outfile,"|\n");

list = list->next;

}break;

Case5:

list\_region(reg);

struct city \*list ;

list =first;

while(list != NULL){

if(strcmp(list->region , region) == 0){

printf("%d %15s %3s %d\n",list->plate,list->cityName,

list->region,list->number\_of\_neigbour);

}

list = list->next;

}

break;

Case6:

list\_number\_of\_neig(ans);

struct city \*list;

list=first;

while(list != NULL){

if(list->number\_of\_neigbour > n){

printf("%d %15s %3s %d\n",list->plate,list->cityName,

list->region,list->number\_of\_neigbour);

}

list = list->next;

}

break;

Case7:

Exit(); fonksiyonlarını çalıştırmaktadır.

•Menü kısmı tamamlandıktan sonra açılan dosyalar kapanmakta ve program sonlanmaktadır.

# Bize Kazandırdıkları ve Sonuçlar

*1)Bağlı Liste Yapısının İyice Kavranması:*

Projede dosyadan okunan değer yapılar üzerinde saklandı ve yapılar bağlı listelere aktarıldı . Bu sayede bağlı listenin etkin kullanılması,üzerinde herhiangi işlem yapabilmek , arama yapılabilmesi sağlanmış oldu.

2) *Dosyadan Formatlı Okuma Yapılabilmesi ve Dosyaya Tekrar Yazılımın Öğrenilmesi:*

Dosyadan istenilen formatta okuma fscanf ile gerçekleştirildi ve gerekli işlmler yapıldıktan sonra tekrar dosyaya yazma işlemlerinin kavranması ve pekiştirilmesi sağlandı.

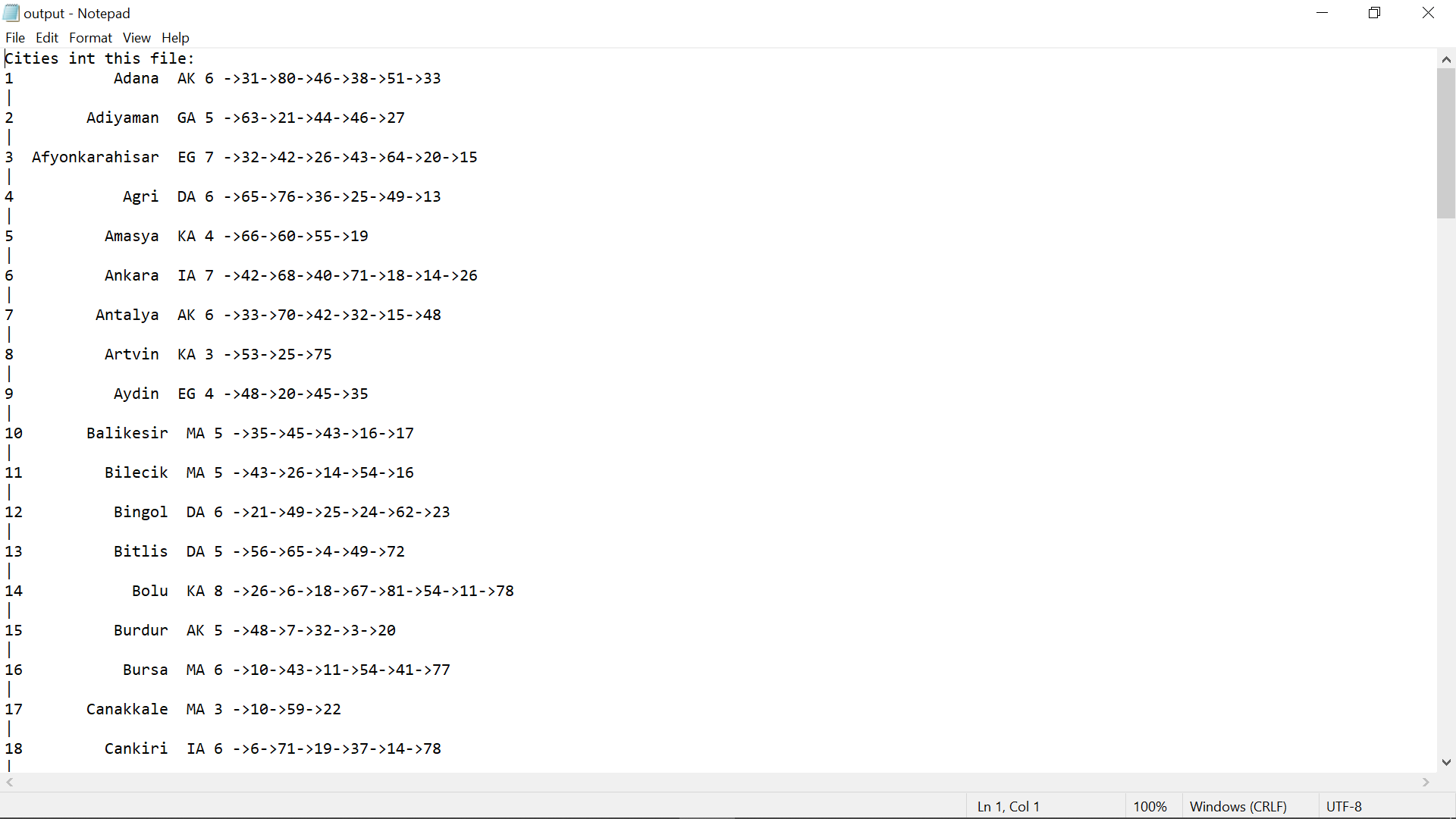
# Kaynakça

•https://www.youtube.com/watch?v=wDAf9Er6Qq8 **-> Sadi Evren Bağlı Liste Uygulamaları**

•https://drive.google.com/drive/folders/1EF1e6nB3I4Z\_PEFCYKMCQ4mzkeqmdFDE->Programlama 2 Drive Dosyaları

•https://www.tutorialspoint.com/index.htm->Java Tutorialpoint for c/c++

***7.Uygulama Görüntüleri:***

******