Muayene Kuyruğu Kontrol Sistemi

Proje Raporu

Konular

Giriş

1.1 Projenin Hedefleri

Projenin ana hedefi muayeneler, hastaneler ve benzeri kurumlarda sıra problemini hal etmek. Bunun için Queue (Kuyruk) veri yapısını kullandım. Her gelen hastaya sıra vermeden İsim, Soyisim ve TC numara alarak otomatik olarak sıraya sokuyoruz ve ilk geleni ilk çağırıyoruz. (FIFO)

1.2 Kullanılan Dil ve Kütüphaneler

Dil olarak özellikle data structures özellikleri ile daha çok avantaj sağladığı için C++ kullanılmıştır.

Kütüphane olaraksa standart input ve output kullana bilmek için <iostream>, dosya işlemleri için <fstream>, getline fonksiyonunu kullana bilmek için ise <string> kütüphanelerini kullandım.

1.3 Kullanılan Araçlar

Kod yazma aşamasında IDE olarak Microsoft Visual Studio kullanılmıştır. Derleyici olarak MinGW-w64 kullanılmıştır.

Kodlar

1.

```
struct Node {
    string ad, soyad;
    long long int tc;

    Node* next;

    Node(long long int t, string a, string s)
    {
        tc = t;
        ad = a;
        soyad = s;
        next = NULL;
    }
};
```

*"Node" isimli bir **struct** yaratıyoruz. Bu **struct** hasta kayıt bilgilerini tutacak. Hastanın Ad ve Soyadını **string** türünde değişkende tutuyoruz (**char** dizisinde de tutula bilirdi) . TC numarasını da tutmak için **long long int** türünde "tc" değişkeni tutuyoruz. Türününün **int** değilde **long long int** olmasının sebebi **int** maksimum 2147483647 sayısını tuta biliyor. TC kimlik numarası ise 11 haneli olduğundan **int** kullanamıyoruz. Öte yandan **long long int*** (2^63)-1 e kadar sayıları tuta biliyor. *O yüzden burada **long long int** kullanıyoruz.*

*Bu veri yapımız tek yönlü kuyruk olduğundan bir tek ondan sonra gelecek **Node**'un adresini "next" değişkeninde tutuyoruz.*

Ve sonda da yapıcı bir fonksiyon yazıyoruz. Aldığı parametreleri değişkenlere atıyor ve sonraki düğümü NULL yapıyor.

2.1.

```
void HastaEkle()
{
    ofstream fout;


    string ad, soyad;
    long long int tc;

    cout << endl << endl << "Hasta Adi:  ";
    cin >> ad;

    cout << "Hasta Soyadi:  ";
    cin >> soyad;

    cout << "Hasta TC:  ";
    cin >> tc;


    // Aldiğimiz bilgilerle yeni Hasta yaratiyoruz.
    Node* temp = new Node(tc, ad, soyad);
```

ofstream *ile dosya akımı değişkeni yaratıyoruz. Sonrasında kullanıcıdan alacağımız değerleri tutması için değişkenler oluşturuyoruz. Hastanın adını, soyadını ve TC numarasını alıp uygun değişkenlere atıyoruz ve sonrasında o değişkenlerle* ***Node*** *türünde bir değişken yaratıyoruz.*

2.2.

```
if (rear == NULL) {
    fout.open("Hasta.txt");
    fout << tc << " |" << ad << " " << soyad <<  endl;
    front = rear = temp;
    return;
}
else
{
    fout.open("Hasta.txt", ios::app);
    fout << tc << " |" << ad << " " << soyad <<  endl;
}

rear->next = temp;
rear = temp;

fout.close();
```

*Sonrasında "rear"-ın NULL olmasını sorguluyoruz. Eğer NULL ise bu kuyruğun boş olduğunu gösteriyor. Dolayısıyla dosyanın en başından yazmaya başlayacağız ve **<tc |ad soyad>** formatında dosyaya verileri kayıt ediyoruz ve kuyruğun başını da, sonunu da bu hasta yapıyoruz çünki sadece 1 hasta var. Ama eğer NULL değilse o zaman dosyayı açarken **app** (append) modunda açıyoruz. Bu sayede dosyaya yazdığımız şeyler dosyada bir şeyi değişmeden sonuna yeni hastayı ekleyecek.*

Sonra kuyruğun sonuna hastamızı ekleyip dosya akımını kapatıyoruz.

3.

```
void ilk_satiri_sil()
{
    ifstream is("Hasta.txt");

    ofstream ofs;
    ofs.open("temp.txt", ofstream::out);

    char c;
    int line_no = 1;
    while (is.get(c))
    {

        if (line_no != 1)
            ofs << c;

        if (c == '\n')
            line_no++;
    }

    ofs.close();
    is.close();

    remove("Hasta.txt");

    rename("temp.txt", "Hasta.txt");
}
```

*Bu fonksiyonda ise **ifstream** ile text dosyasını ilk olarak okuma modunda açıyoruz (is). Sonra 2.1 numaralı kod bloğundaki gibi yeni bir text dosyası açıyoruz. Ardından yine geçici bellek rolü görecek **char** bir değişken yaratıyoruz. Bu değişken Hasta.txt dosyasındaki her bir karakteri almamıza yardım edecek. Hemen altında **int** tipinde "line_no" değişkenini yaratıyoruz. Bu değişken bizim hangi satırda olduğumuzu tutacak. O yüzden başlangıç olarak 1-e eşitliyoruz. **While** döngüsünün içindeki durumda "is" dosya akımından bir karakter alıp "c" değişkenine atıyoruz. Bu döngü atayacak bir karakter bulamayana kadar devam edecek. Döngünün içindeki ilk koşulda ilk satırda olup olmadığımızı kontrol ediyoruz. Eğer ilk satırda değilsek doğrudan aldığımız karakteri temp.txt dosyasına yazdıracağız. Eğer ilk satırdaysa yazdırmıyoruz çünkü silme işleminde FIFO mantığıyla ilk giren ilk çıkmalı. O yüzden ilk satır hariç hepsini temp.txt dosyasına kopyalamamız lazım. İkinci koşulda ise "c" "\n" karakterine eşitse "line_no"-yu bir arttır diyoruz, çünkü "\n" "new line" yani yeni yeni satırı gösteren karakter. Karakterler bitip döngüden çıktığımız zaman her iki açtığımız dosyayı kapatıyoruz. Hasta.txt silip, sonrasında temp.txt dosyasının ismini Hasta.txt olarak değişiyoruz. Böylelikle ilk satır silinmiş oluyor.*

4.

```
void HastaSil()
{
    if (front == NULL)
        return;


    Node* temp = front;
    front = front->next;

    if (front == NULL)
        rear = NULL;

    ilk_satiri_sil();

    delete (temp);
}
```

Önce durum kontrolünde "front"-un NULL olup olmamasını kontrol ediyoruz. Eğer NULL ise bu kuyruğun boş olması demek. Bu durumda silecek bir şey olmadığından fonksiyon boş geri dönüyor. Eğer boş değilse ***Node*** *türünde bir temp değişkeni yaratıp "front"-u orada tutuyoruz. Sonra front u kendinden sonraki elemana eşitliyoruz. Bu sayede en öndeki* ***Node*** *silinmiş oluyor. Eğer işlem sonrası "front" NULL olursa "rear"-ıda NULL yapıyoruz. Ardından dosyada da değişiklikleri kayıt etmek için ilk satırı silme fonksiyonunu çağırıyoruz ve son olarak "temp" i bellekten tamamile siliyoruz.*

4.

```
void Listele()
{
    string line;
    ifstream fin;

    fin.open("Hasta.txt");

    if (fin) cout << endl << endl << "TC          | Isim Soyisim " <<
                            endl << "------------|---------------------" << endl;

    else cout << endl << "Kayit Bulunmamakta.";

    while (fin) {

        getline(fin, line);

        cout << line << endl;
    }

    fin.close();
}
```

İlk olarak ***string*** *türünde bir değişken oluşturuyoruz, bu değişken dosyadan alacağımız satırı ekrana yazdırana kadar tutacak.* ***ifstream*** *ile dosyayı okuma modunda açıyoruz. Koşulumuzda dosyada bir şey olup olmadığını kontrol ediyoruz, eğer varsa tablonun üst kısmını oluşturuyoruz, yoksa ekrana "Kayit Bulunmamakta." yazdırıyoruz. Ardından program* ***while*** *döngüsüne giriyor. Orada da dosyada bir şey olması durumu kontrol ediliyor. Dosya boşalmadıkca döngü dönmeye devam edecek. Dögünün içerisinde dosyadan bir satır alınıp doğrudan ekrana bastırılıyor. Dosya boşaldıktan sonra dosya akımını kapatıyoruz.*

5.

```
int main()
{

    Queue q;
    string* options = new string[4]{    "1. Siraya Hasta Ekle" ,
                                        "2. Siradan Hasta Al",
                                        "3. Sirayi Listele",
                                        "4. Programi Bitir"     };
    int option = 0;

    cout << "||||||||| Hasta Sira Sistemi |||||||||" << endl << endl;


    while (option != 4) {

        cout << endl << endl    << options[0] << endl
                                << options[1] << endl
                                << options[2] << endl
                                << options[3] << endl
                                << endl << "Seciminiz: ";

        cin >> option;

        switch (option)
        {
            case 1:

                q.HastaEkle();
                break;
            case 2:
                q.HastaSil();
                break;
            case 3:
                q.Listele();
                break;
            case 4:
                break;
            default:
                cout << "Yanlis Girdiniz" << endl;
                break;
        }
    }

    return 0;
}
```

***Main** fonksiyonu program koşmaya başladığında ilk çağırılan fonksiyon olduğundan başlangıç için gereken şeyleri burada yapıyoruz. İlk olarak kuyruk yapımızı oluşturuyoruz. Bu bize incelediğimiz üzere bir başlangıç bir de son düğümleri yaratıp NULL-a eşitleyecek. Ardından **string** tipi bir dizide kullanıcıya*

*sunacağımız seçenekleri tutuyoruz. Bir sonra ki satırda kullanıcının seçtiği seçeneği tutmak için **int** tipi bir değişken yaratıp hiçbir karara etki etmemesi için 0 a eşitliyoruz. Menümüzü sürekli tekrarlanan yapa bilmek için kullanıcı 4 seçmediği sürece devam eden bir **while** döngüsü yazıyoruz. Ardından **cout** ile menümüzü ekrana yazdırıyoruz ve kullanıcıdan birini seçmesini istiyoruz. Sonra kullanıcının seçimini "option" değişkenine atıyoruz ve **switch** bloğuna giriyoruz.*

"option" 1 olduğunda HastaEkle(),

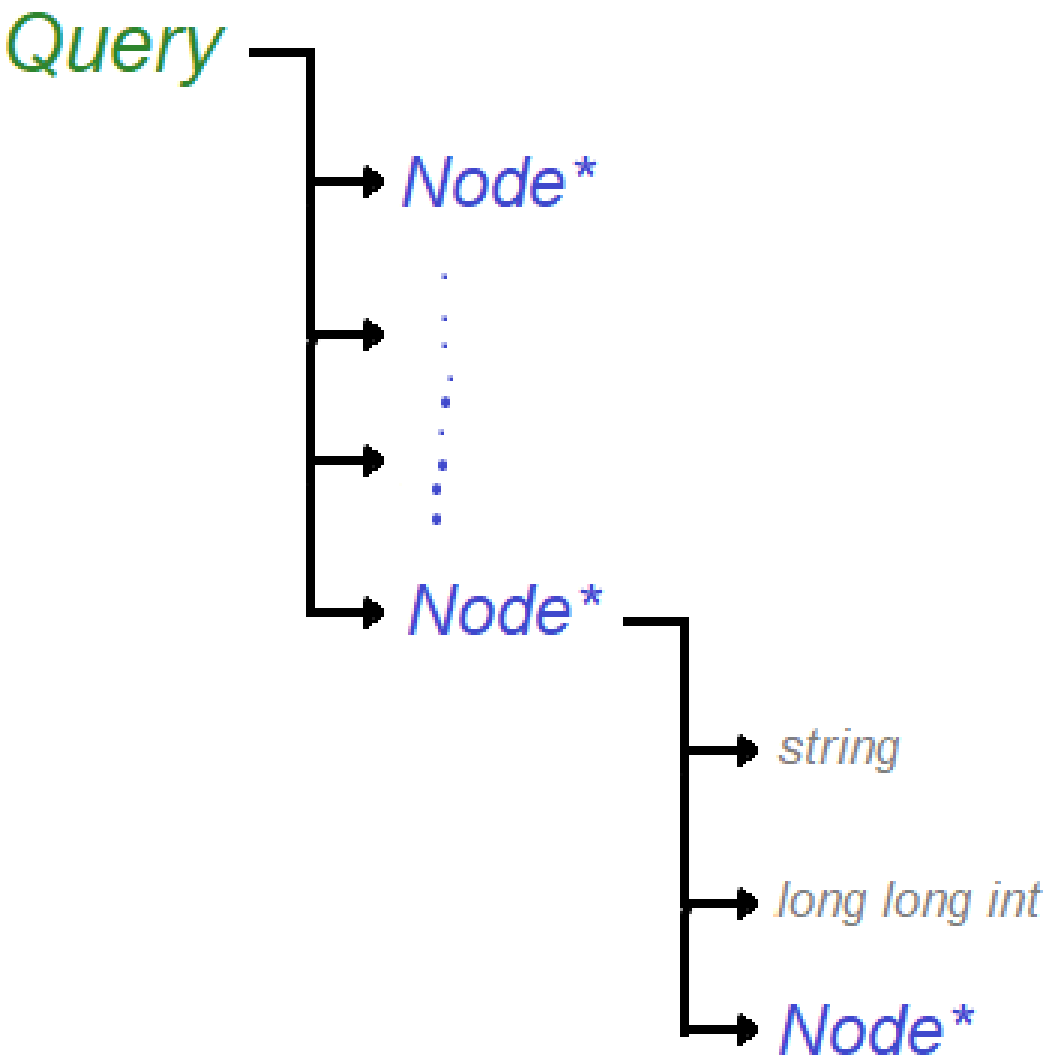
2 olduğunda HastaSil(),

3 olduğunda Listele() fonksiyonunu çağırıyor,

4 olduğunda ise switch den çıkıyor ve option 4 olduğundan while döngüsünden de çıkıyor.

Eğer beklenmedik bir şey girilirse default a gidiyor ve ekrana "Yanlış Girdiniz" yazdırıyor. Her bir case in sonunda break kullanıyoruz, eğer kullanmazsak option 2 olduğunda hem HastaSil(), hem de Listele() fonksiyonunu çağıracak ve ekrana "Yanlış Girdiniz" yazdıracak. En sonda main fonksiyonumuz 0 geri döndürüyor.

2.1 Değişkenler



2.2 Fonksiyonlar

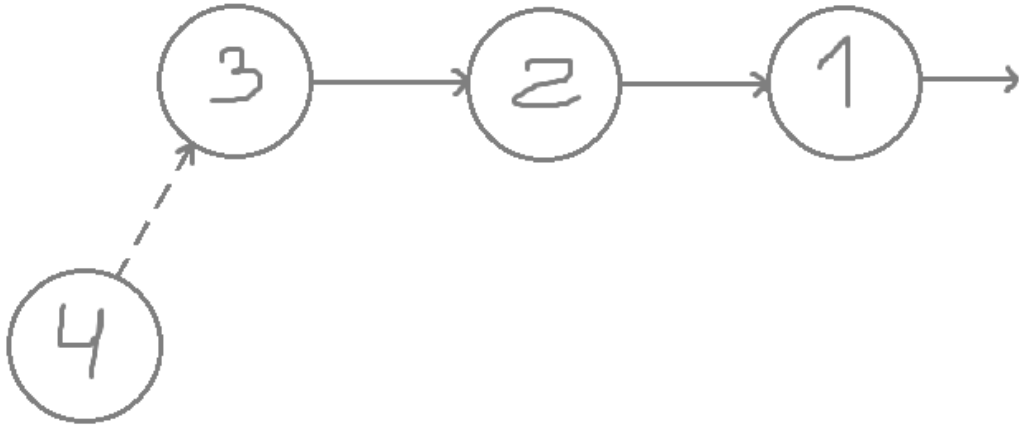
- `Node(long long int t, string a, string s)`

  Node struct'ının yapıcı fonksiyonu olup veriler parametrelerin atamalarını yapıyor.

- `Queue()`

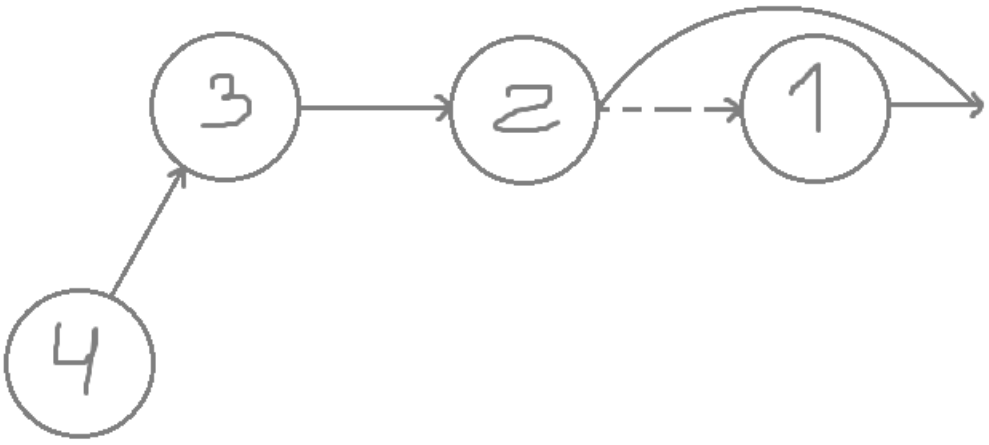
  Queue struct'ının yapıcı fonksiyonu olup kuyruğun başını ve sonunu NULL yapıyor.

- `HastaEkle()`



Bu fonksiyon çağırıldığında ilk olarak 4 düğümü yaratılıyor. Sonrasında mevcut kuyruk yapısının arkasına ekleniyor. Ardından "Hasta.txt" dosyasına da ekleniyor.

- `HastaSil()`



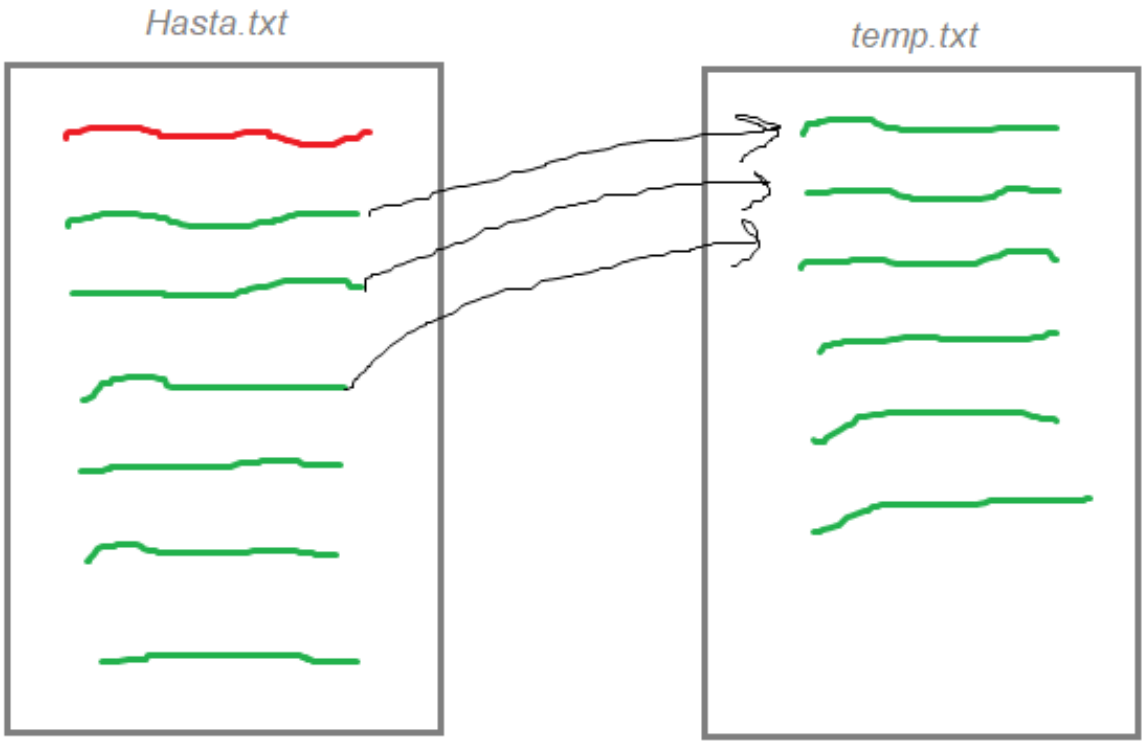
Bu fonksiyonda ise çağırıldı zaman kuyruğun başındaki düğümü(1) bir değişkende

tutup ondan öncekileri sırada bir öne çekiyor. Sonrasında ilk_satiri_sil() fonksiyonu çağırılıyor ve düğüm yok ediliyor.

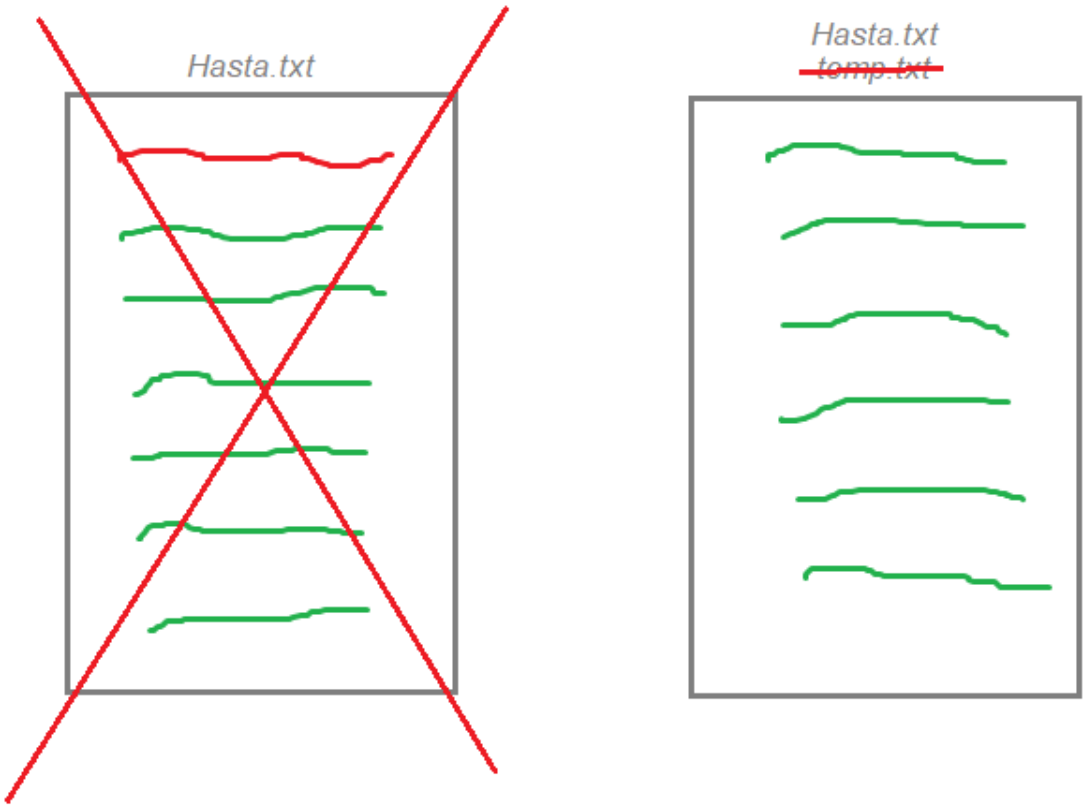
- `Listele()`

  Kuyruktaki bilgilerin yazıldığı dosyadan bilgileri alıp ekrana yazdırıyor.

- `ilk_satiri_sil()`



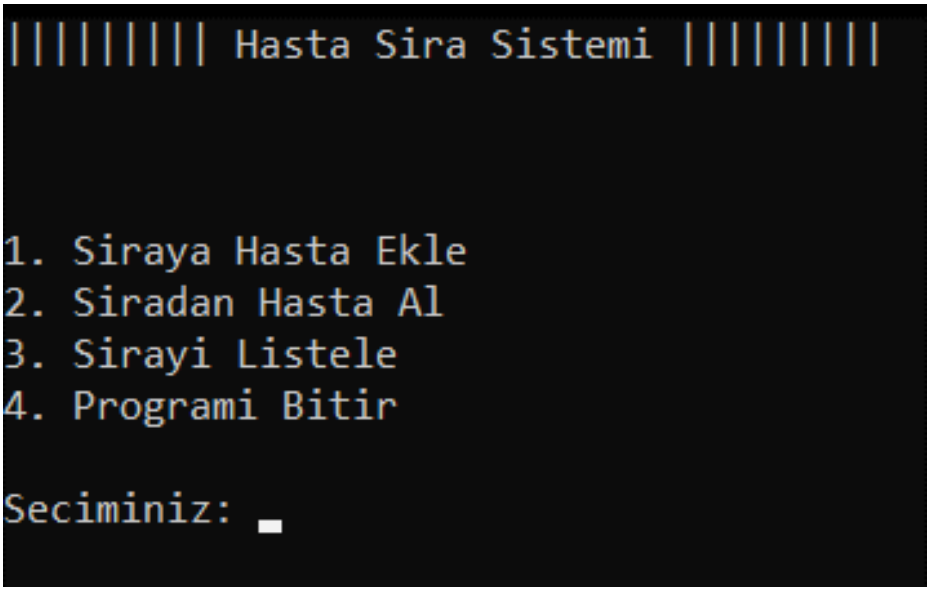
İlk olarak Hasta.txt dosyasında ilk satır hariç hepsini temp.txt dosyasına yeniden yazdırıyoruz. Sonrasında da Hasta.txt dosyasını siliyoruz ve temp.txt dosyasının adını değişip Hasta.txt koyuyoruz. Böylelikle ilk satır silinmiş olur.



- `main()`

  Ana Menüyü ekrana yazdırıyor ve girilen seceneğe göre fonksiyonlar çağırıyor.

2.3 Ekran Görüntüleri



Ana Menü

Hasta Ekleme

```
Seciminiz: 3


TC          | Isim Soyisim
------------|---------------------
12345678909 |Serife Tan
98765432112 |Merve Ozturk
11111111111 |Cem Yilmaz
```

Kuyruğun Listelenmesi

```
Seciminiz: 2


1. Siraya Hasta Ekle
2. Siradan Hasta Al
3. Sirayi Listele
4. Programi Bitir

Seciminiz: 3


TC          | Isim Soyisim
------------|---------------------
98765432112 |Merve Ozturk
11111111111 |Cem Yilmaz
```

Sıradan Hasta Almak

Referanslar

https://www.geeksforgeeks.org/queue-linked-list-implementation/

https://stackoverflow.com/questions/9505085/replace-a-line-in-text-file

https://www.tutorialspoint.com/cplusplus/cpp_data_types.htm