Ad : Fırat

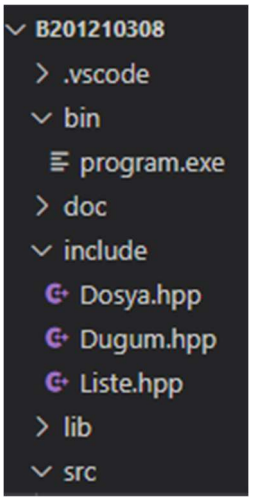
Soyad : Turan

Öğrenci Numarası : B201210308

Grup : 2.Öğretim B gurubu

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

Veri Yapıları Ödev-1



Yapmış olduğum projede program, açıldığı gibi Veri.txt dosyasını okuyacak ve her satırda yapılacak işlemi belirten bilgiyi okuyup ekleme ve silme işlemlerini gerçekleştirecek. Veriler iki yönlü bağıl listeye eklenecek fakat bu iki yönlü bağıl liste dizi gibi davranacak.

Sol taraftaki görselde görüldüğü gibi 3 farklı başlık oluşturdum.

Dosya başlığı altında Veri.txt dosyasının okunması ve bu dosya içerisindeki E(2#Fırat Turan) ve S(3) gibi bilgileri satır bazında kontrollerini sağladım.

Dugum başlığı altında iki yönlü bağıl listedeki düğümlerin yapısını tasarladım.

Liste başlığı altında oluşturduğumuz iki yönlü bağıl listeye eleman ekleme, silme ve listeleme gibi işlemlerini gerçekleştirdim.

Dugum

```
#ifndef Dugum_hpp
#define Dugum_hpp
#include<iostream>
#include<string>
using namespace std;

class Dugum
{
public:
    Dugum(string veri);
    string veri;
    Dugum* sonraki;
    Dugum* onceki;
};

#endif
```

Oluşturmuş olduğum Dugum class'ı içine:

string veri; → Düğümün içinde tutacağı veri.

Dugum* sonraki; → Bu düğümden sonraki düğümün adresi.

Dugum* önceki; → Bu düğümden önceki düğümün adresi.

İki yönlü bağıl liste oluşturmak istediğimiz için yukarıdaki gibi bir tanımlama yaptık.

Tanımlamış oluğumuz düğümün önceki ve sonraki değerlerini 0 'a eşitledik.

Liste

```
#ifndef Liste_hpp
#define Liste_hpp
#include<exception>
#include<string>
#include "Dugum.hpp"

class Liste
{
public:
    Liste();
    void ekle(int sira,string veri);
    void cikar(int sira);
    friend ostream& operator<<(ostream& os,Liste& Liste);

private:
    Dugum* dugumGetir(int sira);
    Dugum* ilk;
    Dugum* son;

};

#endif
```

void ekle(int sira,string veri); →Bağlı listeye eleman eklemek için oluşturduk.”sira” eklemek istediğimiz sıra. “veri” eklenecek veri. Listenin sonuna yeni bir düğüm ekledik. Seçilen düğümden(“sira”) sağa doğru verileri kopyaladık. Son olarak seçilen düğüme “veri”‘yi kopyaladık. Eğer belirtilen “sira” numarasında bir indeks yok ise listesinin sonuna ekleme yaptık.

void cikar(int sira); → Bağlı listeden eleman silmek için kullandık.”sira” silmek istediğimiz indeks. Seçilen düğüme sonraki düğümün verisi gelecek şekilde listenin sonuna kadar kopyaladık. Son olarak listenin sonundaki veriyi sildik. Eğer belirtilen “sira” numarasında bir indeks yok ise listenin sonundaki elemanı sildik.

Dugum* dugumGetir(int sira); → Verilen “sira” değişkenindeki indekste bulunan verinin adresini geri döndüren bir metot oluşturduk. Bu metodu bağlı listeye ekleme ve çıkarma yapmak için oluşturduğum ekle ve çıkar metotlarında kullandım.

Dugum* ilk; → Bağlı listedeki ilk veriyi tutan bir değişken tanımladım.

Dugum* son; → Bağlı listedeki son veriyi tutan bir değişken tanımladım.

Dosya

```
#ifndef Dosya_hpp
#define Dosya_hpp
#include<string>
using namespace std;

class Dosya
{
public:
    Dosya();
    void dosyaOku(string dosya);
private:
    string veriBul(string line);
    int indeksBul(string line);
};
#endif
```

void dosyaOku(string dosya);→Veri.txt dosyasının içinde bulunan verileri satır bazında okumasını sağladık. dosya değişkenine txt dosyasının isim bilgisini atadık.

string veriBul(string line);→ dosyaOku metodundan satır bazlı okuduğum veriyi “line” değişkenine atadık. Atamış olduğumuz satırdaki veriden düğüme eklenecek veriyi ayıkladık. Örneğin E(2#Fırat Turan) şeklinde bir verimiz olsun bu veriden “Fırat Turan” bilgisini geri gönderdik.

int indeksBul(string line);→ dosyaOku metodundan satır bazlı okuduğum veriyi “line” değişkenine atadık. Atamış olduğumuz satırdaki veriden indeks bilgisini ayıkladık. Örneğin E(2#Fırat Turan) veya S(3) şeklinde verilerimiz olsun bu verilerden sırasıyla “2” ve “3” değerlerini döndürdük.