TELEFON KAYIT SİSTEMİ

BLM 102 ALGORİTMA VE PROGRAMA I

2019-2020 BAHAR DONEMİ PROJE

- Projenin Amacı
- Tasarım
- Kullanılan Araçlar
- Proje Ekibi

PROJENİN AMACI

Bir telefon rehberinde bulunması gereken kayıt ekleme, kayıtları gösterme, kayıtları modifiye etme, kayıt arama ve silme kabiliyetlerine sahip olan bir sistem geliştirilecektir. Geliştirilmiş sistem, kullanıcıya komut satırı ara yüzü sunarak rehberin kullanımını sağlayacaktır.

TASARIM

Tasarıma kullanıcıdan aldığımız bilgileri saklayacağımız değişkenler ve bunların listeleneceği vektörleri tutan ‘kişi’ isimli sınıfı yaratarak başladık.

```
class kişi
{public:
    int Id;
    string ad;
    string soyad;
    long long cep;
    long long iş;

    vector <int> ID_vec;
    vector <string> ad_vec;
    vector <string> soyad_vec;
    vector <long long> cep_vec;
    vector <long long> iş_vec;
};
```

İkinci bir sınıf olarak ‘telefon’ adlı sınıf üretilmiştir. Bu sınıfa kişi sınıfından kalıtım yapılmıştır . Kişi sınıfındaki değişkenler ve vektörler kullanılarak bunlar üzerinde verileri sıralama , değiştirme , arama ve silme gibi çeşitli operasyonları gerçekleştirebilmek için telefon sınıfında bu kabiliyetlere sahip fonksiyonlar üretilmiştir.

```
class telefon:public kişi
{
public:
    telefon();
    void kayit();
    void kayitGoster();
    void kayitModifiye();
    void arama();
    void kayitSilme();


};
```

MENÜ TASARIMI

- Kullanıcının istediği operasyonu yapabilmesi için kaynak dosyamızda bir menü tasarladık.

```
int main()
{int secim=0;
telefon t1;
t1.Id = 0;
while (secim != '6')
{
    cout << "\n....:::::::TELEFON REHBERI MENU::::::::....\n";
    cout << endl;
    cout << "1. Yeni kayit ekle.\n";
    cout << "2. Kayit listesini goster.\n";
    cout << "3. Kayit degistir.\n";
    cout << "4. Arama yap.\n";
    cout << "5. Kayit sil.\n";
    cout << "6. Cikis yap.\n";
    cout << "**Secim yapiniz: ";
    cin >> secim;
```

```
    if (secim == 1)
    {
        t1.kayit();

    }
    if (secim==2)
    {
        t1.kayitGoster();
    }
    if (secim == 3)
    {
        t1.kayitModifiye();
    }
    if (secim == 4)
    {
        t1.arama();
    }
    if (secim == 5)
    {
        t1.kayitSilme();
    }
    if (secim == 6)
    {
        break;
    }
}
```

REHBERE KAYIT EKLEME

- Kullanıcıdan yeni ekleyeceği kişiye ait bilgiler gerekli değişkenlere kayıt edilmiştir . Daha sonra bu bilgiler yine gerekli vektörlere depolanmak ve çeşitli işlemlerde kullanılmak üzere gönderilmiştir.

```
void telefon::kayit()
{   Id++;
    cout << "\n.....::KAYIT SISTEMI::.....\n";
    cout << endl;
    cout << "Kayit edilecek kisinin ismi: ";
    cin >> ad;
    cout << "Kayit edilecek kisinin soyismi: ";
    cin >> soyad;
    cout << "Kayit edilecek kisinin numarasi: ";
    cin >> cep;
    cout << "Kayit edilecek kisinin is numarasi: ";
    cin >> iş;



    ID_vec.push_back(Id);
    ad_vec.push_back(ad);
    soyad_vec.push_back(soyad);
    cep_vec.push_back(cep);
    iş_vec.push_back(iş);



}
```

REHBERDEKİ KAYITLARI GÖSTERME

- Kullanıcıdan alınan kayıtlar alfabeye göre sıralanmış biçimde ekrana bastırılmıştır.

```
void telefon::kayitGoster()
{
    cout << "\n.................::KAYIT LISTESI::.................\n";
    cout << endl;
    for (int i = 0; i < ad_vec.size(); i++)
    {
        for (int j = 0; j < i; j++)
        {
            string ad=ad_vec[i];
            string ad2 = ad_vec[j];
            if (ad.at(0) < ad2.at(0))
            {
                int temp_int;
                string temp_string;
                long long temp_long;

                temp_int=ID_vec[i];
                ID_vec[i] = ID_vec[j];
                ID_vec[j] = temp_int;

                temp_string = ad_vec[i];
                ad_vec[i] = ad_vec[j];
                ad_vec[j] = temp_string;

                temp_string = soyad_vec[i];
                soyad_vec[i] = soyad_vec[j];
                soyad_vec[j] = temp_string;

                temp_long = cep_vec[i];
                cep_vec[i] = cep_vec[j];
                cep_vec[j] = temp_long;

                temp_long = iş_vec[i];
                iş_vec[i] = iş_vec[j];
                iş_vec[j] = temp_long;

            }
        }
    }
```

```
ofstream dosya;
    dosya.open("rehber.txt", ios::out);

    for (int i = 0; i < ad_vec.size(); i++)
    {
        if (ID_vec[i] != 0)
        {   dosya << "ID:" <<ID_vec[i] << " Adi:" << ad_vec[i] << " Soyadi:" << soyad_vec[i] << " Cep.No.:" << cep_vec[i] << " Is.No.:" << iş_vec[i] << endl;
            cout<<"ID:" << ID_vec[i] <<" Adi:"<< ad_vec[i] << " Soyadi:" << soyad_vec[i] << " Cep.No.:" << cep_vec[i] << " Is.No.:" << iş_vec[i] << endl;
        }
    }
    dosya.close();
}
```

KAYITLARI MODİFİYE ETME

Kullanıcıdan alınan her kayda eşsiz birer ID numarası verilmiştir . Kayıt değiştirme işlemi değiştirilmek istenen kaydın ID numarası ile başlar ve her bir değişken için kullanıcıdan yeni girişler beklenir . Kullanıcının var olmayan bir ID girmesi durumunda try-catch mekanizmasıyla kullanıcı uyarılır.

```
void telefon::kayitModifiye()
{
    cout << "\n..................::KAYIT MODIFIYE SISTEMI::..................\n";
    cout << endl;
    cout << "degistirmrk istenilen kaydin ID'sini giriniz:";
    int searchID;
    cin >> searchID;
    cout << endl;

    int i;
    int flag = 0;
    try {
        for (i = 0; i < ID_vec.size(); i++)
        {
            if (ID_vec[i] == searchID)
            {
                flag = 1;
                break;
            }
        }
        if (flag == 1)
        {
            cout << "yeni isim:";
            cin >> ad_vec[i];
            cout << "yeni soyisim:";
            cin >> soyad_vec[i];
            cout << "yeni cep no:";
            cin >> cep_vec[i];
            cout << "yeni is no:";
            cin >> iş_vec[i];
        }
        else
            throw "ID'ye sahip kisi bulunamamistir.";
    }
    catch (const char* e)
    {
        cout << e << endl;
    }
```

```
ofstream dosya;
dosya.open("rehber.txt", ios::out);

for (int i = 0; i < ad_vec.size(); i++)
{
    if (ID_vec[i] != 0)
    {
        dosya << ID_vec[i] << " " << ad_vec[i] << " "
            << soyad_vec[i] << " " << cep_vec[i] << " " << iş_vec[i] << endl;

    }
}
dosya.close();
```

KAYIT ARAMA

Kullanıcının isteği doğrultusunda kişinin istenilen özelliğine göre arama yapılmaktadır . Kullanıcının arama kriteri kayıtlarla uyuşmaması durumunda kullanıcı uyarılmaktadır.

```
void telefon::arama()
{
    cout << "\n.....................::KAYIT ARAMA SISTEMI::.....................\n";
    cout << endl;

    cout << "ID->1" << endl;
    cout << "isim->2" << endl;
    cout << "soyisim->3" << endl;
    cout << "cep no->4" << endl;
    cout << "is no->5" << endl;
    cout << "Aram kriteri seciniz:";
    int tercih;
    cin >> tercih;
    cout << endl;
    if (tercih ==1)
    {
        cout << "ID'ye gore arama yapiliyor" << endl;
        int searchID;
        cin >> searchID;
        cout << endl;
        int i;
        int flagID = 0;
        try {
            for (i = 0; i < ID_vec.size(); i++)
            {
                if (ID_vec[i] == searchID)
                {
                    flagID = 1;
                    break;
                }

            }
            if (flagID == 1)
            {
                cout << ID_vec[i] << " " << ad_vec[i] << " " << soyad_vec[i] << " " << cep_vec[i] << " " << iş_vec[i] << endl;
            }
            else
                throw "Arama kriterinize uygun kisi bulunamadi !";
        }
        catch (const char* e)
        {
            cout << e << endl;
        }
    }
```

KAYIT SİLME

Silinmek istenen kaydın ID ‘si alınır . Eşleşen ID varsa kayda ait veriler silinir eğer ID ye ait kayıt yoksa kullanıcı uyarılır.

```
void telefon::kayitSilme()
{
    cout << "\n.................::KAYIT SILME SISTEMI::.................\n";
    cout << endl;
    cout << "silinecek kaydin ID'sini giriniz:" << endl;

    int searchID;
        cin >> searchID;
        cout << endl;
        int i;
        int flagDelete = 0;
        try
        {
            for (i = 0; i < ID_vec.size(); i++)
            {
                if (ID_vec[i] == searchID)
                {
                    flagDelete = 1;
                    break;
                }
            }
            if (flagDelete == 1)
            {
                ID_vec[i] = 0;
                ad_vec[i] = ' ';
                soyad_vec[i] = ' ';
                cep_vec[i] = 0;
                iş_vec[i] = 0;
                cout << "kayit silinmistir !" << endl;
            }
            else
                throw "silmek istediginiz ID'ye sahip kisi bulunamadi!";
        }
        catch (const char* e)
        {
            cout << e << endl;
        }
```

```
ofstream dosya;
dosya.open("rehber.txt", ios::out);
for (int i = 0; i < ad_vec.size(); i++)
{
    if (ID_vec[i] != 0)
    {
        dosya << ID_vec[i] << " " << ad_vec[i] << " " << soyad_vec[i] << " " << cep_vec[i] << " " << iş_vec[i] << endl;

    }
}
dosya.close();
```

KULLANILAN ARAÇLAR

PROJE EKİBİ

- Hasan Demir 190301004
- Berk Tunç 190301003
- İlhan Ersoy 190301024