T.C.

SAKARYA ÜNİVERSİTESİ

BİLGİSAYAR VE BİLİŞİM BİLİMLERİ FAKÜLTESİ

BİLGİSAYAR MÜHENDİSLİĞİ BÖLÜMÜ

PROGRAMLAMA DİLLERİNİN PRENSİPLERİ ÖDEV RAPORU

ÖDEV-2

Grup Elemanları:

[redacted] - **1/A Ömer Can ÇALIŞIR**

[redacted] - **1/A Melih Ensar BARIŞIK**

SAKARYA

Nisan, 2020

Özet

Programda bizden istenilen rastgele kişi oluşturabilecek bir kütüphane oluşturmamızdır. Kişi sınıfı özellikleri ise şunlardır: T.C. kimlik, isim, soy isim, yaş ve telefon. Bunlara ek olarak telefon özelliğine dahil olarak bir imei özelliği bulunmaktadır. Kişi özelliklerinde T.C. kimlik numarası ve imei numarası belirli bir algoritmaya sahiptir ve oluşturulurken iken bu algoritmaya bağlı olacak şekilde doğru biçimde oluşturulmuştur. İstenilen kişiler oluşturulduktan sonra bir metin belgesinde kayıt edilir. Kayıt edilen bu kişi verileri sırası ile tekrar farklı bir işlem için geri okunur. Okunan verilerden T.C. kimlik numarası ve telefon özelliğine dahil olan imei numarası bilgileri kontrol edilir. Çıktı olarak bu bilgilerin doğru ya da yanlış olduğu ekrana yazdırılır. Bütün özellikleri oluştururken kendi yazmış olduğumuz rastgelelik sınıfından yararlandık.

Anahtar kelimeler: Fonksiyon göstericisi, nesne yönelimli programlama paradigması, nesne yönelimli benzetim, Java-C dilleri arasındaki farklar

1.Geliştirilen Yazılım

Kısım 1 Java

Projemizin yapımına kullanılacak sınıfları tasarlayarak başladık. Bu sınıfların; Control sınıfı, Rastgele sınıfı, Kisi sınıfı, IMEINo sınıfı, KimlikNo sınıfı, RastgeleKisi sınıfı, ReadFile sınıfı, Telefon sınıfı, WriteFile sınıfı, Test sınıfı olmasına karar verdik.

Tasarımımızın uygulamasına kişi sınıfını oluşturarak başladık. Ödevde belirtildiği üzere KimlikNo ve Telefon referansı, isim soy isim ve yaş değişkenleri oluşturduk. Bu alanların kapsülleme ilkesine bağlı olarak get ve set metotlarını oluşturduk. Dışarıda oluşturup Kisi sınfına gönderilen değerleri Kisi sınıfının kurucu fonksiyonu ile alıp değişkenlere atama işlemi yaptık. Kişi sınıfının tasarımı bu şekilde tamamlanmış oldu.

Sonrasında başka fonksiyonlarda kullanmak üzere istediğimiz zaman rastgele rakam ve rastgele sayı üretebilen Rastgele sınıfı tasarımına başladık. Ödevin birince kısmında sayı üretebilen produceRandomDigit fonksiyonunu tasarladık. ProduceRandomDigit fonksiyonu rastgele sayı hesaplarken sistmeden okunan nano saniye ve paramtere olarak aldığı limit değerini kullanır. Önce tekrarı azaltmak için nano saniye yüze bölünür daha sonra istenilen limit ile modu alınarak verilen limite uygun bir rastgele sayı döndürür. İkinci fonksiyonumuz olan produceRandomNumber fonksiyonu verilen uzunlukta sayı üretmek için parametre olarak aldığı decimalPlaces ve ilk fonksiyonumuz olan produceRandomDigit yardımı ile rastgele sayı üretmektedir. Parametre olarak verilen uzunluk değeri kadar produceRandomDigit fonksiyonunu çağırıp içerisine rakam elde etmek için on sayısını verir ve geri dönen rakamları yan yana birleştirerek döndürür. Rastgele sınıf tasarımını böylelik ile bitirdik.

Her kişiye özel olan T.C. kimlik numarasını üretmek için KimlikNo sınıfını kullandık. Kimlik numaralarını üretirken algoritmaya sadık kaldık. Oluşturulacak olan kimlik numarasını tutmak için string tipinde bir identityNumberS değişkeni ve rastgele fonksiyonlardan rakam üretmek için integer bir limit değişkeni tanımlayıp bu değişkene on değerini atadık. İlk fonksiyon olan, firstStepControlAndConstitute fonksiyonunda rastgele sınıf yardımı ile bir rakam ürettik. Üretilen bu rakamın algoritmada belirtildiği üzere sıfır olmamasına özen gösterdik. İkinci aşama olan constituteIdentityNumber fonksiyonunda rastgele sekiz basamaklı bir sayı üretip ilk üretilen rakama ekledik. CalculateTenthStep fonksiyonunda ise algoritmaya göre sırası ile tek ve çift basamakları kullanarak onuncu basamağı hesapladık. Hesaplamada tek basamakları toplayıp 7 ile çarptık bu çarpımdan çift basamakları çıkararak on sayısı ile modunu aldık ve onuncu basamağı oluşturmuş olduk. Son basamağı yani kontrol basamağını hesaplamak için ise calculateEleventhNumber fonksiyonu içinde ilk on basamağı toplayıp on ile modunu alarak son basamağımızı elde etmiş olduk. Daha sonra ileride Control sınıfında da kullanmak üzere controlIdentityNumber fonksiyonunu yazdık. Bu fonksiyon kısaca verilen kimlik numarasının uzunluğunun 11 olmasına, ilk basamağın 0 olmamasına, algoritmada istenilen üzere ilk 9 basamağın tek basamaklarının yedi katı ile çift basamaklarının farkını on ile modunun onuncu basamağı vermesine ve son olarak bu on basamağın toplamını on ile modunun, on birinici yani son basamağı verip vermemesine göre doğru ya da yanlış sonucu döndürmektedir. CreateIdnetityNumber fonksiyonunu ise adımların sırası ile çalıştırılması için yazdık. Hesaplanan kimlik numarası değerini kullanmak için kapsülleme prensiplerine uygun olarak getKimlikNo fonksiyonun yazdık ve KimlikNo sınıfını tamamladık.

IMEINo sınıfı oluşturulan numarayı tutmak için bir IMEIS isimli bir string değişkene sahiptir. İmei numarasını oluşturmak için ise ilk 14 rastgele rakamımızı constituteImeıNoPart1 fonksiyonunda üretiyoruz. Algoritmaya sağdık kalacak şekilde on beşinci rakamı hesaplamak için ise calculateRecentDigit fonksiyonunda tek sayıların kendilerinin çift sayıların ise iki ile çarpılmış hallerinin basamaklarını topluyoruz. Sayıların basamaklarını toplamada collectNumbers fonksiyonunda yazdığı yapı sayesinde parametre olarak alınan çift sayıların iki katının rakamları toplanıp geri çevrilir. Bu toplamın kendinden önceki en büyük onun katı olan sayı ile farkı bize on beşinci basamağımızı veriyor. Üretilen bu son basamağı da ekleyince imei üretme işlemimiz tamamlanıyor ve createImei fonksiyonu ise bu fonksiyonların sırası ile çağırılmasında görev alıyor. ControlImei fonksiyonumuz verilen bir imei numarasının on beş basamaklı olmasına ve ilk on dört hanesini kullanarak kontrol hanesi olan on beşinci haneyi doğru hesaplayıp hesaplamadığına bakıyor. Oluşturulan imei numarası getImeiNo ile kapsüllenerek geri döndürülüyor.

Telefon sınıfında ise oluşturulan telefon numarasını tutmak üzere bir telefonNo string değişkeni ve referans olarak oluşturulan IMEINo1 referansı bulunur. Bunun sebebi her telefon numarasına ait bir imei numarası olmasıdır. Bunun yanı sıra içerisinden rastgele seçim yapmak için bir string operatorler değişkenimiz vardır. Telefon sınıfımızın kurucusu içerisinde bahsi geçen IMEINo1 referansına oluşturulan IMEINo nesnesi atanır. Doğru bir telefon numarası üretmek için telefonVeIMEINoUret isimli fonksiyonda telefonNo

değişkeninin ilk hanesine sıfır değeri atandıktan sonra operatorler içerisinden rastgele şekilde bir alan kodu çekilir. Oluşan telefon numarasının üstüne telefon numarasının uzunluğunu sağlamak için 7 adet rastgele rakam eklenir.Ayrıca bu fonksiyonumuz ile ürettiğimiz telefon numarasına ait olan imei numaramızı üretmek için IMEINo sınıfında tanımladığımız createImei fonskiyonu da burada çağırılır. Kapsülleme prensipleri gereği sınıf dışında kullanılması gereken telefonNo ve IMEINo1 değişkenleri için geri döndürme yani get fonksiyonları yazılır. Bu şekilde Telefon sınıfımızı tamamlarız.

ReadFile sınıfında ise verilen dosyayı okuyup içerisindeki bilgileri satır satır ve kelime kelime tutan bir fonksiyon bulunmaktadır. Satır sayısını bilmediğimiz dosyanın bilgilerini satır satır tutmak üzere, dinamik bir dizi olan informationsLinebyLine ArrayList'ini tanımladık. Aynı şekilde içerisindeki kelime sayısını bilmediğimiz dosyanın bilgilerini kelime kelime tutmak üzere, dinamik bir dizi olan informationsWordsbyWords ArraList dizisini tanımladık. Daha sonra parametre olarak okuyacağı dosyanın adını alan ReadFile isimli bir fonksiyon tanımladık ve gövdesini yazmaya koyulduk. Öncelikle dosyayı satır satır okuyup bir değişkene atadık. Daha sonra split fonksiyonu yardımıyla while içerisinde her satır sonunda eklediğimiz satır sonu anlamına gelen \n ifadesine göre satırları böldük ve kelimeDizisi adlı diziye atadık. Yani sonuç olarak KelimeDizisinin her bir dizininde dosyadaki satırları saklamış olduk. Ardından bu dizinde dönerek bilgileri informationsLinebyLine değişkenimize atadık. Bu işlemin ardından bu sefer bilgilerin kullanımı kolaylaştırmak üzere ikinci bir split kullanımıyla, bilgileri kelime kelime tutan informationsWordsbyWords değişkenine atadık. Dosya işlemleri yapıldığından dolayı bir hatayla karşılaşılması durumunda hatayı yakalamak adına bu işlemleri try-catch bloğu içerisinde gerçekledik. Son olarak informationsLinebyLine ve informationsWordsbyWords dizileri için kapsülleme prensiplerine uygun olarak gerekli get metodlarını yazdık. Böylelikle ReadFile sınıfının tasarımı tamamlanmış oldu.

RastgeleKisi sınıfında temel amacımız Kisi sınıfından üretilen nesneleri daha sonra yazma işleminde kullanmak için depolamaktır. Bu amaç için öncelikle oluşturulacak kişileri depolamak için kisiler adında bir ArrayList oluşturuldu. İşlemi yapan kisiUret fonksiyonunda ReadFile sınıfından bir nesne oluşturarak okumak istediğimiz dosyanın adını bu sınıftaki ReadFile nesnesine verdik. Üretilmek istenen kişi sayısı kadar dönen bir for tanımlayarak üretme işlemine başladık. KimlikNo sınıfından bir nesne oluşturup gerekli fonksiyon ile bir kimlik numarası ürettik. Rastgele sınıfından bir okunan metnin satır sayısı limitinde bir sayı üretip okuduğumuz metinden rastgele isim ve soy isim aldık. Kişinin yaşı için limit yüz olacak şekilde rastgele sayı ürettik. En son telefon nesnesini de üretip tüm bilgiler ile beraber bir kişi nesnesi oluşturduk. Oluşan bu nesneyi ArrayListimize ekledik. Aynı sınıfa dahil getKisiler nesnesi bize kişileri istediğimiz yere iletmemizde yardımcı oldu ve biz de yine aynı sınıfa dahil olan dosyaYaz fonksiyonuna kişileri ayrı bir metin belgesine yazdırmak için verdik.

WriteFile sınıfında usingPrintWriter isimli bir fonksiyon tanımladık. Bu fonksiyon dışarıdan RastgeleKisi sınıfında üretilen kisi ArrayListini ve açılacak olan dosya adını parametre olarak almaktadır. Dosya açıldıktan sonra kisi nesnesi içindeki tüm bilgilere erişilerek tek bir değişken olan bilgiler stringine toplanır ve bu şekilde dosyaya yazılır. Dosyaya yazarken imei numaramızın başına ver sonuna parantez işareti ekliyoruz. Dosyaya yazma işlemi her çalıştığında bir önceki veriler dosyadan silinir.

Control sınıfında ise temel amaç yazılan verilerden istenenler yani imei numarası ve kimlik numarasının okunup doğru bir şekilde yazılıp yazılmadığını kontrol etmektir. Bu amaç için 2 adet ArrayList tanımladık bunların isimleri identityNumbers ve imeiNumbersdır. İlk fonksiyonumuz chooseInformationToCheck ArrayList dizisi olan words yardımı ile kelime şeklinde okunan kişi bilgilerini depoluyoruz. Depolanan kelimelerin ilk sırasına kimlik bilgilerini ve altıncı sırasına imei numaralarını yazdığımız için verileri okurken buna dikkat ediyoruz. Imei yazarken kullandığımız parantezleri ise substring yardımı ile ayıklayıp sadece rakamsal değeri elde ediyoruz. Elde edilen rakamsal değerleri tanımladığımız ArrayListlere kontrol amaçlı kullanmak için atıyoruz. Dosya okuma işleminde yanlışlık olmaması için ayrıca try catch kullanıyoruz. Okunan verileri kullanmak için controlIdentityNumber ve controlImeiNumber isimli iki fonksiyonumuz bulunuyor. Bu kısımda depoladığımız kimlik ve imei numaralarını sınıflarında bulunan kontrol fonksiyonlarına gönderiyoruz. Önce rakamsal değerimizi daha sonra ise değerin algoritmaya uygun olup olmamasına göre doğru yanlış bilgilerini bastırıyoruz.

Test kısmında öncelikle hangi işlemin istendiğini öğrenmek için bir menü yazdırdık. Gelen cevap bir ise RastgeleKisi sınıfından bir nesne oluşturduk ve oluşturulmak istenen kişi sayısını kullanıcıdan öğrenerek istenilen sayıda kişi ürettik. Daha sonra dosyaYaz fonksiyonu ile oluşan kişiler dosyaya yazıldı. Gelen cevap iki ise Control sınıfında bir nesne oluşturup kontrol için gerekli olan fonksiyonları sırası ile çağırdık. Cevap üç ise program durdurulur ve çıkış yapılır cevap bunlardan farklı ise program aynı ekranı tekrar karşımıza getirildi.

Kısım2 C

Sınıf tasarımı kısım birde yaptığımız ile aynıdır.

Tasarımımızın uygulamasına kişi kütüphanesini oluşturarak başladık. Ödevde belirtildiği üzere KimlikNo ve Telefon referansı, isim soy isim ve yaş değişkenleri oluşturduk. Ayrıca ileride tanımlayacağımız fonksiyon için uygun fonksiyon göstericisi tanımladık. Typedef yapısı ile yapımızın kullanımını kolaylaştırdık. Kütüphane içerisinde herhangi bir gövde yazma işlemi yapılmadı.

Dışarıda oluşturup Kisi yapısına gönderilen değerleri Kisi yapısını sözde kurucu fonksiyonu ile alıp değişkenlere atama işlemi yaptık. Sözde kurucu fonksiyonumuz KisiOlustur içerisinde ilk olarak Kisi yapısından oluşturulacak her kişi için bellekten malloc işlemi ile gerekli alan ayrıldı. Dışarıdan alınan değerler Kisi yapısındaki değişkenlere atandı. Bu fonksiyonumuz içinde malloc ile her bir kişi için ayırdığımız alanı serbest bırakan yokediciKisi fonksiyonunun adresi fonksiyon göstericisine atandı. YokediciKisi fonksiyonu parametre olarak aldığı kişiyi bulunduğu bellek adresinden silen fonksiyondur.

Tasarımımıza Rastgele kütüphanesi ile devam ettik. Kütüphane tasarımında RASTGELE yapısı içinde kullanmak üzere sadece pozitif değerler alan bir integer tanımladık. Daha sonra fonksiyon göstericilerini tanımlayarak yapıyı tamamladık. Typedef yapısı ile yapımızın kullanımını kolaylaştırdık. Sözde kurucumuz RastgeleOlustur, asıl işlemi yapan _randomRakamUret fonksiyonumuzu ve yıkıcı fonksiyon görevini üstlenen _yokediciRastgele fonksiyonunun gövdelerini burada tanımladık.

RastgeleOlustur fonksiyonu içerisinde bu sınıftan bir nesne oluşturup malloc ile bellekte ihtiyacı olan alanı ayırdık. Kütüphanemizde tasarlamış olduğumuz integer değere burada time.h kütüphanesi yardımı ile bir sayısal değer ataması yaptık. Son olarak fonksiyon göstericilerine gerçek fonksiyonların adreslerini atadık. Rastgele üretme işlemini yapan _randomRakamUret fonksiyonumuz dışarıdan bağlı olduğu yapının sözde nesnesini ve oluşturulabilecek sayı aralığını aldı. Alınan bu aralık ile sözde kurucumuzda ataması yapılan pozitif integer değerimiz çeşitli matematiksel işlemlere tabi tutularak istenilen aralıkta bir random sayı üretmiş olduk._yokediciFonksiyonumuz sözde kurucumuz içinde ayrılan bellek alanını serbest bırakmak görevini yapmaktadır.

KimlikNo kütüphanesinde ise KIMLIKNO yapısı içerisinde oluşturulacak olan kimlik numarasını tutmak için integer değer oluşturduk. İleride hesaplamak için kullandığımız fonksiyonların fonksiyon tutucularını tanımladık. Typedef yapısı ile kullanımı kolaylaştırdık.

KimlikNoUret fonksiyonu içerisinde bu kütüphaneden bir sözde nesne oluşturup ihtiyaç duyacağı alanı malloc ile sağladık. Kimlik uzunluğu olan on bir değerini uzunluğa verip gerekli alanı kimlik değeri için de sağladık. Bir döngü yardımı ile ilk değerlerimizin hepsine sıfır atadık. Daha sonra fonksiyon tutucularına fonksiyon adresleri atandı ve bu fonksiyon tamamlanmış olduk. _firstStepControlAndConstitute fonksiyonu kimlik numarasının ilk değerini verme işini yapar. Kimlik numarasının ilk değeri sıfır olamayacağı için buna dikat ediliyor. _constituteIdentityNumber fonksiyon ile de ilk rakamdan sonra gelecek sekiz rakamın rastgele ataması yapılıyor. _calculateTenthStep fonksiyonunda ise kısım1de detaylarını açıkladığımız kimlik numarası oluşturma algoritmasını göz önünde bulundurarak onuncu basamağı hesapladık. Kontrol ve son basamak olan on birinci basamağı _calculateEleventhNumber fonksiyonu ile hesapladık. Hesaplama işlemelerini yaparken kısım1de detaylarını açıkladığımız algoritmaya sağdık kaldık. Tüm bu oluşturma fonksiyonlarını sırası ile çağırmak için bir _createIdentityNumber fonksiyonu tanımladık. ControlIdentityNumber fonksiyonu ise dışarıdan verilen kimlik numarasının algoritmaya uygun olup olmamasına göre doğru ya da yanlış değeri çevirmektedir. En son _yokediciKimlikNo fonksiyon yardımı ile sözde kurucu içinde ayırdığımız bellek adreslerini temizleme işlemi yapmaktadır.

IMEINo kütüphanesinde bir IMEINo yapısı oluşturduk. Oluşturulan bu yapı içerisinde bir tanesi imei numarasının kendisi ve diğeri ise uzunluğu olmak üzere iki adet integer değer tanımladık. Değerleden sonra fonksiyon göstericilerimizi tanımladık. Typedef yapısını kullanım kolaylığı için kullandık.

IMEINoUret fonksiyonunda oluşturulan yapı için malloc ile bir değer açıldı. Kütüphanede tanımlanan ime numaramıza ilk değer ataması yapıldı ve fonksiyon göstericilerine fonksiyon adresleri verilerek tamamlandı. _assignRandomDigits fonksiyonunda imei numaramızın ilk on dört değeri rastgele bir şekilde atandı. _calculateRecentDigit fonksiyonumuzda ise amacımız daha önce atanan ilk on dört değeri kullanarak on beşinci değeri hesaplamaktır. Kısım1de de belirttiğimiz gibi araştırdığımız algoritmaya uygun bir şekilde hesaplama yaparken _collectNumbers fonksiyonunu kullanıyoruz. Bu fonksiyonun amacı parametre olarak verilen değeri alıp basamaklarındaki değerlerin toplamını hesaplamaktır. Üretim işlemini bu sırada sağlayabilmek için bir _createImeiNo fonksiyonuna sahibiz. Üretilen bu sayıları ya da parametre olarak veirlen herhangi bir sayının imei numarası üretme algoritmasına uygun üretilip üretilmediğini kontrol eden bir controlIMEI fonksiyonumuz bulunmaktadır. Son olarak sözde kurucuda ayırılan alan _yokediciIMEINo fonksiyonumuz ile serbest bırakılmaktadır.

Telefon kütüphanesinde bir TELEFON yapısı oluşturduk. Bu yapı içerisinde oluşturulacak olan telefon numarasını tutmak üzere bir integer dizi, uzunluk değerimizi saklamak için bir integer değeri ve her telefonun sahip olduğu imei numarasını depolamak için bir IMEINo referansı tanımladık. Fonksiyon göstericilerinin tanımlanması ile bu yapı tamamlanmış oldu.

TelefonOlustur sözde kurucu fonksiyonu ile ilk olarak ihtiyacımız olan alanı bellekten malloc yardımı ile ayırdık. Uzunluk değerimiz olan dokuzu atadık. Dokuz değerini atama sebebimiz operator alan kodunu tek bir integer değer içinde tutuyor olmamızdır. Burada kütüphanede tanımladığımız imei nesne referansını gerçekledik. Telefon numarasına ilk değer ataması yaptıktan sonra fonksiyon göstericilerine fonksiyonların adreslerini vererek bu fonksiyonu tamamladık. _telefonVeIMEINoUret fonksiyonumuz her telefon numarasında olduğu için ilk değere sıfır ataması yaptık. Daha sonra alan kodlarını içeren operatörler integer dizisi içinden rastgele şekilde bir operatör seçerek ikinci değerimize bunu atadık. Burada dizimizin bir bölgesine üç rakam atadığımız için uzunluğumuz normal telefon numarasından kısa olmaktadır. Geriye kalan numara kısmına ise rastgele rakamlar atadık. Nesne referansı olarak çağırılan imei numaramız

için gerekli fonksiyon olan _createImei fonksiyonunu çağırarak imei numaramızı da üretiyoruz. En son olarak _yokediciTelefon içerisinde önce üretilen imei nesne referansı daha sonra Telefon yapısının tuttuğu alanları temizliyoruz.

ReadFile kütüphanemizde READFILE isimli bir yapı oluşturarak dosyamızdan okuyacağımız isimleri depolamak için bir iki boyutlu char dizisi oluşturuyoruz. Daha sonra fonksiyon göstericilerimizi tanımlıyoruz.

ReadFileOlustur fonksiyonumuz ile ilk önce gerekli olan alanı malloc ile oluşturuyoruz. Fonksiyon göstericilerine fonksiyon adreslerini atıyoruz. _DosyaOku fonksiyonumuz verilen dosyayı alıp fgets ile satır satır okuyup yapı içinde tanımladığımız iki boyutlu diziye atıyor. Dizide depolanan isimlerden biri rastgele seçilip _getSelectedNameSurname fonksiyonu ile kullanılacağı yere döndürülüyor. Son olarak _yokediciReadFile sözde kurucu içinde bellekten ayırılan alanı serbest bırakılıyor.

RastgeleKisi kütüphanesinde RASTGELEKISI isimli bir yapı oluşturup bu yapı içerisinde daha sonra oluşturacağımız kişileri depolamak için bir kişi işaretçisi oluşturduk. Fonksiyon göstericileri tanımladık.

RastgeleKisiOlustur fonksiyonumuz parametre olarak kaç kişi oluşturulduğunu alıyor. Alınan bu sayıya göre kişi işaretçimize yeterli alanı malloc ile sağlıyoruz. RastgeleKisi nesnesi oluşturup ona da malloc ile gerekli alanı tedarik ediyoruz. Fonksiyon göstericilerine fonksiyonların adresleri veriliyor. _kisiUret fonksiyonu ile okuma ve yazma işlemleri için nesneler üretiyoruz daha sonra bir döngü yardımı ile üretilecek kişi sayısı kadar tekrar işlemi yapıyoruz bu tekrar işlemi içerisinde kişiyi oluşturan özelliklerin oluşturulması ve hesaplanması için fonksiyonlar çağırılıyor. Bu işlemden sonra oluşturulan kişiler tekrar bir döngü yardımı ile yazma işlemine gönderiliyor. Fonksiyon kapanmadan önce oluşturulan okuma yazma nesneleri siliniyor. _yokediciRastgeleKisi fonksiyonu ile sözde kurucuda ayrılan alan temizleniyor. YokediciKisileriSil fonksiyonu içinde oluşturulan kişileri ve kişilerin sahip olduğu telefon ve kimlik numarası gibi bellekte yer kaplayan alanlar temizleniyor.

WriteFile kütüphanesinde WRITEFILE yapısı içerisinde fonksiyon göstericileri tanımlanıyor.

OlusturWriteFile fonksiyonumuz içerisinde öncelikle gerekli olan alanı bellekten malloc ile ayırıyoruz ve fonksiyon göstericilerine fonksiyonların adreslerini atıyoruz. Asıl işlemin yapıldığı _writeKisiFile fonksiyonunda ise parametre olarak kişi ve yazılmak istenilen dosya alınıyor. Alınan kişi üzerinden içerisinde depoladığı bilgiler sırası ile verilen dosyaya yazılıyor. Yazma işlemi kullanıcı her yeni kişi sayısı verdiğide önceki veriler silinerek yapılıyor. _yokediciWriteFile fonksiyonu sözde kurucuda ayrılan bellek alanını serbest bırakıyor.

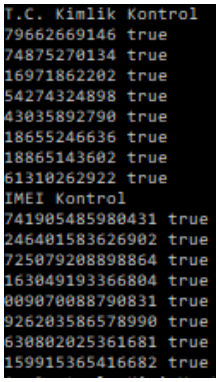
Control kütüphanesinde kontrol etme işleminde kullanılacak işlemlerin ve yıkıcı fonksiyonun fonksiyon göstericileri tanımlanıyor.

ControlOlustur fonksiyonu ile bellekten yapı için malloc yardımı ile yer ayrılıyor. Fonksiyon göstericilerine fonksiyonların adresleri atanıyor. _chooseInformationToCheck fonksiyonunda önce okunacak dosya açılıyor. Okunacak değerleri tutmak için uygun diziler oluşturduk. Oluşturulan bu dizilere fscanf yardımı ile okunan dosyalar her boşluk ayrım şartı olmak üzere okunuyor ve dizilerde depolanıyor. Daha sonra ilk olarak controlIdentity fonksiyonumuz çalışıyor bu fonksiyon parametre olarak oluşturulan kimlik numarası dizisini ve uzunluğunu alıyor. Alınan dizi içerisinde bir döngü yardımı ile dönülerek döngü içerisinde KimlikNo sınıfında bulunan controlIdentityNumber fonksiyonuna sırası ile gönderiliyor. Önce sayısal değer daha sonra ise doğru yanlış olup olmadığı yazdırılıyor. _controlImeiNumber fonksiyonun da bir fark bulunuyor. Bu fark imei numarasının başında ve sonunda bulunan parantez işaretlerinden kaynaklanıyor. Char olarak aldığımız bu değerleri işlem yapmak için integer tipine çevirip rakamsal değerini alıyoruz. Daha sonra IMEINo sınıfında bulunan controlIMEI fonksiyonuna gönderip önce rakamsal değeri daha sonra ise doğru ve yanlış olma olayını yazıyoruz. Son olarak _yokediciControl fonksiyonu ile sözde kurucu içinde ayrılan alanı serbest bırakıyoruz.

Mainde öncelikle bir Rastgele sözde nesnesi oluşturuyoruz. Daha sonra kullanıcıya bir menü gösterip yapmak istediği işlemin numarasını yazmasını istiyoruz. Yazılan sayı bir ise öncelikle üretilecek kişi sayısını kullanıcıdan alıyoruz ve sözde RastgeleKisi nesnesi oluşturup sözde kurucuda işlemleri başlatıyoruz. Daha sonra oluşturulan sözde nesne ile kisiUret fonksiyonunu çağırıp istenilen sayıda kişinin üretilip ayrı bir metne yazılma işlemini başlatıyoruz. İşlemler bittikten sonra oluşturulan sözde RastgeleKisi nesnesini kapatıyoruz. Cevabımız iki ise dosyada yazılı olan kimlik ve imei numarası bilgilerini kontrol edip ekrana yazdıran Control sınıfında bir sözde nesne oluşturup işlemleri başlatıyoruz işlemlerin bitişinde yıkıcı fonksiyon çalışıyor. Cevap üç ise oluşturulan Rastgele fonksiyonu siliniyor ve program kapanıyor. Verilen cevap bunlardan biri değil ise menü ekranı tekrar karşımıza geliyor.

Kullanılan Yöntemler Ve Amaçları

Rastgele sayılar üretirken hazır kütüphaneler kullanmak yerine kendimiz oluşturmaya çalıştığımızda birçok sorunla karşılaştık. Bunların çözümü için internetteki kaynaklardan yararlandık. Ödevin verilme amaçları arasında bizi bu araştırmaya teşvik etmek olabilir. Ayrınca nesne yönelimli programlamayı destekleyen ve desteklemeyen dilleri kullanarak bu dillerin arasındaki farkları görmüş olduk ve nesne yönelimli programlamanın işleri ne kadar kolaylaştırdığını görmüş olduk. C de nesne yönelimli programlamaya benzetmek için struct yapısı, fonksiyon gösterici, sözde kurucu fonksiyon gibi yöntemlere başvurduk. Ayrıca kod yazımını kolaylaştırmak için typedef yöntemini kullan



Algoritma Adımları

1. Üretilecek kişi sayısı alınır

2. İsim ve soyisim dosyadan çekilir

3. Yaş ve kimlik numarası, telefon numarası ve telefona ait imei numarası üretilir.

4. Kişi üretilir

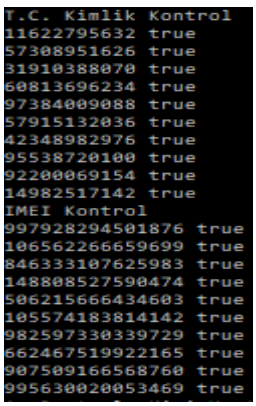
5. Oluşturulan kişiler dosyaya yazılır.

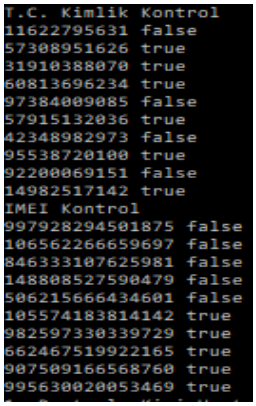
6. İstenildiği taktirde dosyadan okuma yapılır.

7. Seçili bilgiler kontrol edilir.

8. Programdan çıkılır

2. Çıktılar





Menüde bir tuşuna basıldığı zaman kullanıcının verdiği sayı kadar kişi üretiliyor ve üretilen bu kişiler dosyaya yazılıyor. Daha sonra iki tuşuna basılınca yukarı görüldüğü gibi kişilerin kimlik ve imei numaraları kontrol edilip doğru ya da yanlış oldukları ekrana yazılıyor. İki tuşuna her basıldığında dosyada olan eski verilerin üstüne yeni veriler yazılıyor.

3. Sonuç

Nesne yönelimli programlamanın sağladığı avantajları ödevin ilk kısmını yazarken gördük. Dil nesne yönelimli olduğu zaman işlemlerin daha kolay gerçeklendiğini farkettik. Aynı işlemleri kısım ikide yaparken nesne yönelimli desteği olmamasına rağmen nesne yönelimli paradigmaya benzeterek kısım birde yaptığımız gibi gerçeklemeye çalıştık. Bu bağlamda fonksiyon gerçekleme, sözde kurucu fonksiyon, sözde nesne ve işaretçinin önemini görmüş olduk.

Referanslar

https://stackoverflow.com/

https://www.quora.com/

https://www.geeksforgeeks.org/

https://www.tutorialspoint.com/index.htm