

LUHN ALGORİTMASININ C DİLİNDE UYGULANMASI

Programlama Dillerinin Prensipleri

ÖDEV 2

Aleyna DİKAL

B191200001

SAKARYA ÜNİVERSİTESİ

Bilişim Sistemleri Mühendisliği

İÇİNDEKİLER

1. Projenin Konusu	2
2. Luhn Algoritması	2
2.1. Luhn Algoritmasının Çalışma Mantığı	2
3. Projenin Yapılma Aşaması	3
4. Proje Hakkında.....	3
5. Ekran Görüntüleri	3
KAYNAKÇA	8

1. Projenin Konusu

Projenin içeriğinde Luhn Algoritması kullanılarak verilen kredi kartı numaralarının doğru formatta olup olmadığı kontrol edilmeye çalışılmıştır. Bunun yanında projenin içeriğinde birçok kodlama tekniği de yer almaktadır. Bunlara örnek vermek gerekirse dosya okumaktan dosya içeriği hakkında kontrol yapmaya. Oluşturulan pointerların bellekte çöp oluşturmaması için silinmesinden dosya içeriğinin kapatılmasına kadar birçok yöntem önceki dönemlerde görülen derslerin tekrarı niteliğinde olmuştur.

2. Luhn Algoritması

Kredi kartı ile alışveriş yapmak istediğiniz zamanlarda bir çoğumuz kart numaramızı yanlış gireriz. Örneğin kart numaranızdaki 44371 yerine 44317 yazarsanız sistem sizden numaranızı tekrarlamanızı ister. Bu hatanın tespiti Luhn Algoritması sayesinde olur. Bir banka kartı numarasının geçerliliği için gerekli olan test, 1954'te IBM'de çalışan bir bilim insanı olan Hans Peter Luhn tarafından, çoğu insanın banka veya kredi kartı olmadan önce icat edildi.

“Modulus 10 Algoritması” olarak da bilinen Luhn Algoritması, bir kullanıcı tarafından sağlanan kimlik numarasının doğru olup olmadığını belirlemek için kullanılan bir formüldür. Formül, kredi kartı numaralarının yanı sıra devlet Sosyal Güvenlik Numaraları (SSN'ler) gibi diğer numara dizilerinin doğrulanmasında yaygın olarak kullanılmaktadır.

Günümüzde Luhn Algoritması, elektronik ödeme sisteminde önemli bir bileşendir ve tüm büyük kredi kartları tarafından kullanılmaktadır.

Luhn Algoritması, kötü niyetli saldırılara karşı koruma sağlamak için değil, yanlışlıkla yapılan hatalara karşı koruma sağlamak için tasarlandı. Algoritmanın işleyişi, 19. yüzyılın başlarında Carl Friedrich Gauss tarafından geliştirilen matematiksel bir teknik olan modüler aritmetik üzerine kuruludur.

2.1. Luhn Algoritmasının Çalışma Mantığı

Şimdi Luhn Algoritması nasıl çalışır bir örnek ile açıklayalım,

Örnek kredi kartı numaramız 1322 4231 4332 2312 olsun ve bu numara üzerinden Luhn Algoritmasını işletelim.

Şimdi bu 16 haneli kredi kartı numarasının 2.rakamı 4.rakamı 6.rakamı diye giderek 16.rakama kadar bu şekilde çift sayılı sıralarda bulunan rakamları toplayalım.

$3+2+2+1+3+2+3+2=18$ rakamını bulduk. Şimdi geriye kalan tek sayılı sıralarda kalan numaraları tek tek 2 ile çarparak toplayalım.

$(1 \times 2) + (2 \times 2) + (4 \times 2) + (3 \times 2) + (4 \times 2) + (3 \times 2) + (2 \times 2) + (1 \times 2) = 2 + 4 + 8 + 6 + 8 + 6 + 4 + 2 = 40$ buradan çıkan sonuçta da 40 rakamını bulmuş olduk. İlk adımda bulduğumuz sayı ile ikinci adımda bulduğumuz sayıyı topladığımızda 58 sonucunu elde edeceğiz. 58 sayısının 10'a göre modundan kalan 0 etmediği için bu kredi kartı numarasının sahte veya hatalı olduğu sonucuna ulaşabiliriz.

2.2. Luhn Algoritması Avantaj ve Dezavantajları

Luhn algoritması herhangi bir tek basamaklı hatayı ve ayrı zamanda bitişik basamakların neredeyse tüm durumlarını algılayacaktır. Bunun yanında İki basamaklı durumlarda örnek vermek gerekirse 04-40 sıfır şeklinde numaraların kombinasyon edildiği

durumlarda Luhn Algoritması istenildiği şekilde çalışmayacak ve burada sahte oluşturulmuş veya hatalı girilmiş numarayı algılayamayacaktır.

Aynı şekilde ikiz numaralarda örneğin 22 yerine 55 yazılması gibi durumlarda algoritma bu hatayı bulamayacak ve bu durumda da istenilen sonucu veremeyecektir.

Burada diğer bir durum ise sayıların 0 ile doldurulması durumudur 0 ile doldurulan sayılar Luhn Algoritması üzerinden rahatlıkla geçebilirler çünkü sıfırların toplamalarının modu yine sıfır olacaktır. Bu da algoritmanın yanılmasına neden olacaktır.

Ancak bu durumlara rağmen algoritmanın çok kolay uygulanabilmesi kolay çalışması gibi etkenlerden dolayı sadece hata kontrolü için kullanılabilir. Bunun yanında ödeme sistemlerinde ekstra kontrollerin bu algoritma dışında yapılması gerekmektedir.

3. Projenin Yapılma Aşaması

Proje yapılmaya başlanmadan önce algoritma olabildiğince anlaşılmalı çalışılmıştır. Algoritmanın anlaşıldığı fark edildikten sonra ilk olarak klasör yapısı ödevin pdf dokümanında istenildiği şekilde oluşturulmuştur. Bunun yanında makefile dosyası klasör yapısının ve dosyaların oluşturulmasının hemen sonrasında sürekli olarak projeyi çalıştıracığımız için ilk olarak oluşturulmuştur. Bu sayede projenin çalıştırılması esnasında sadece ilgili makefile çalıştıran kodu CMD üzerinden çalıştırmak yeterli olmuştur. Ve bu sayede zaman kazanılması amaçlanmıştır.

Bunun sonrasında ilk olarak internet üzerinden C dilinde dosya okuma işlemleri ile ilgili araştırmalar yapılarak main fonksiyonu içerisinde dosya okuma ile ilgili denemeler yapılmıştır. Bu denemelerin sonucunda satır satır okunan değerler üzerinde gerekli işlemleri yapacak olan fonksiyonlar ihtiyaca göre yazılamaya başlanmıştır.

Bunun yanında oluşan problemlere karşı dile ait her şey hatırlanmadığı için stackoverflow gibi kaynaklardan hatalara ilişkin araştırmalar yapılmıştır.

Projeye ilişkin algıtmada her adım olabildiğince bir fonksiyona yaptırılacak şekilde yapılmaya çalışılmıştır. Burada her satırı okunan txt dosyası bir while döngüsü içine alınarak her satıra ilişkin problem ilgili fonksiyonları çağırarak çözülmüştür.

4. Proje Hakkında

Proje içerisinde yapılan testler sonucunda istenilen ödevde dair hiçbir eksiğin olmadığı görülmüştür. Ödev farklı farklı durumlar ile test edilmiş ve edilen kontroller algoritma elle işletilerek de test edilmiştir bunun sonucunda bir eksiğin olmadığı görülmüştür.

5. Ekran Görüntüleri

```
#include <stdio.h>
#include <string.h>
#include <stdlib.h>
```

Proje İçerisinde Eklenen Kütüphaneler

```
// her ikinci basamagin degerini iki katina cikarttigim fonksiyon
int* ikinciBasamagiIkiyleCarp(int* satirDizi, long int uzunluk) {
    // bir satirin uzunlugu kadar donecek dongu
    for(int i = 0; i<uzunluk; i++) {
        // sondan baslayarak her ikinci elemanın degerini 2 ile carptim
        if(i%2 == 1){
            satirDizi[uzunluk-i-1] = satirDizi[uzunluk-i-1] * 2;
        }
    }
    return satirDizi;
}
```

Çift Basamak Değerlerini 2 ile Çarpan Fonksiyon

```
// Eger sayi 9'dan buyukse sayilari topladigim fonksiyon
void sayilariTopla(int* satirDizi, long int uzunluk) {
    for(int i = 0; i<uzunluk; i++) {
        if(satirDizi[i] > 9){
            // 2 ile carptigimiz herhangi biri 2 basamakli ise rakamlarini topladim.
            satirDizi[i] = satirDizi[i] % 10 + 1;
        }
    }
}
```

Sayılar 9 Dan Büyükse Toplayan Fonksiyon

```
// Elde edilen sayilarin hepsini topladigimiz fonksiyon
int tumBasamaklariTopla(int* satirDizi) {
    int toplam = 0;
    // Her bir satirda boşluklar silinince 16 adet rakam oldugu icin 16 kez donecek olan dongu
    for(int i = 0; i<16; i++){
        // Dizideki her bir elemani toplam degiskeni üzerinde topladığımız yer.
        toplam += satirDizi[i];
    }
    return toplam;
}
```

Elde Edilen Sayıları Toplayan Fonksiyon

```
// Toplamin 10'a göre modunu alip gecerli mi gecersiz mi diye kontrol ettigimiz fonksiyon
void kartNumarasiKontrol(int toplam) {
    if(toplam % 10 == 0){
        printf(" - gecerli");
        printf("\n");
    }
    else{
        printf(" - gecersiz");
        printf("\n");
    }
}
```

Kart Numarasının Geçerli Olup Olmadığını Kontrol Eden Fonksiyon

```
// String'te yer alan bosluk karakterlerini sildigimiz fonksiyon
char *boslukKarakterleriSil(char *charDizi) {
    int i = 0;
    int j = 0;
    while (charDizi[i]!='\0') // bos degilse
    {
        if (charDizi[i] != ' '){
            // Bosluk karakteri haricinde olanlari char dizisine ekler bu sayede bosluklar silinmis olur.
            charDizi[j++] = charDizi[i];
        }
        i++;
    }
    // Char dizisinin bittigini göstermek icin eklenmis bos karakter elemani
    charDizi[j] = '\0';
    return charDizi;
}
```

String İçerisinde Yer Alan Boşluk Karakterlerinin Silindiği Fonksiyon

```
// string'teki satir karakterlerini sildigim metod
void satirBosluklariniSil(char* charDizi) {
    int i = 0;
    int j = 0;
    // Dizinin elemani bos karakter olmayana kadar devam edecek olan dongu
    while (charDizi[i]!='\0')
    {
        // satir karakteri ile karsilasirsa diziye almayacak
        if (charDizi[i] != '\n'){
            charDizi[j++] = charDizi[i];
        }
        i++;
    }
    charDizi[j] = '\0';
}
```

Satır Boşluklarının Silen Fonksiyon

```
// Satirdaki karakter sayisi kadar donecek ve satirlari ekrana bastiricak olan fonksiyon
void satirlariYazdir(long int uzunluk,int* satirDizi){
    for(int i=0;i<uzunluk;i++) {
        if(i %4 == 0 && i != 0){
            printf(" ");
        }
        printf("%d", satirDizi[i]);
    }
}
```

Satırları Ekrana Bastıran Fonksiyon

```

int main(int argc, char *argv[]) {
    // Dosya işlemleri için oluşturulmuş File tipinde pointer değişkeni
    FILE* dosyaPointer;

    // Okuduğumuz satırları atacağımız int tipinde satirDizi arrayi
    int* satirDizi;

    /* Satir uzunluğu olarak varsayılan bir deger atadım
    bu değeri seçerken her satırın uzunluğunun 19 olmasından ve 1 adet de boşluk karakteri
    okuyacağı için 21 i seçtim
    döngüler 0 dan başlıyor bu nedenle 19+1 sonrasında 1 adet '\0' karakteri 21 karakter ediyor.
    */
    int satirUzunluk = 21;

    // Her bir satırı temsil edecek string için bellekte yer ayrılması işlemi.
    char* satir = (char*) malloc(satirUzunluk*sizeof(char));

    /* Döngülerde kullanılmak üzere tutulan uzunluk değişkeni başlangıç değeri bu şekilde atanmıştır ancak daha sonradan
    while döngüsü içerisinde satirin uzunluğuna göre yeniden değer alacaktır. Çok fazla karakter alma durumu için long int
    olarak oluşturulmuştur.
    */
    long int uzunluk = 2048;

    /* Dosyanın okuma işlemi için açılması burada verilen ilk parametre dosyanın yoludur ikinci parametre ise
    read modda açıldığını gösterir.*/
    dosyaPointer = fopen("doc/okunacak.txt", "r");
}

```

Main Fonksiyon Ekran Görüntüsü-1

```

// Dosyayı satır satır okuyacak while döngüsü
while(fgets(satir, satirUzunluk, dosyaPointer)) {

    satirBosluklariniSil(satir);

    //Satırda olan boşluk karakterlerini silen fonksiyonun çağırılması
    boslukKarakterleriSil(satir);

    // char dan int dönüşümü yapan fonksiyonun çağırılması
    satirDizi = charDonusumInt(satir);

    // uzunluk degiskenini dongulerde kullanmak icin tuttum. bir satirdaki rakam sayisini tutuyor
    uzunluk = strlen(satir);

    // satirdaki karakter sayisi kadar donecek ve satirlari tek tek yazdiracak
    satirlariYazdir(uzunluk,satirDizi);

    // PDF de bize verilen algoritmanın ilk adımı
    ikinciBasamagiIkiyleCarp(satirDizi,uzunluk);

    // PDF de bize verilen algoritmanın ikinci adımı
    sayilariTopla(satirDizi,uzunluk);

    // PDF de bize verilen algoritmanın üçüncü adımı
    int basamaklarinToplami = tumBasamaklariTopla(satirDizi);

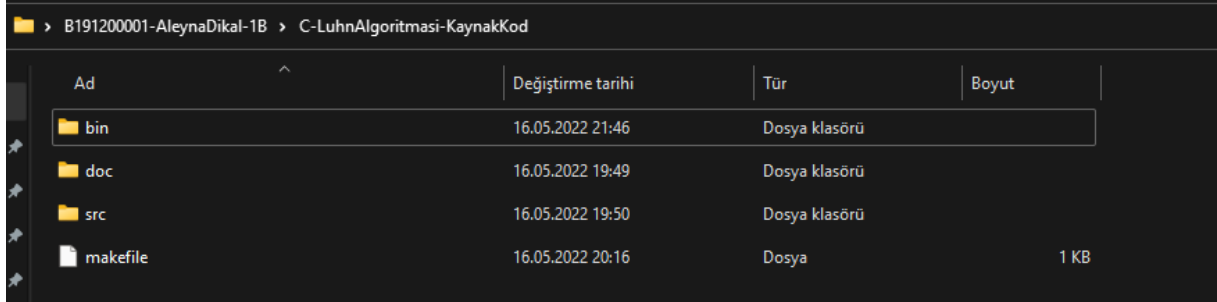
    // PDF de bize verilen algoritmanın dördüncü adımı
    kartNumarasiKontrol(basamaklarinToplami);
}

//Açılan Dosya Okuma İşleminin Kapatılması
fclose(dosyaPointer);

//Oluşturulan pointerların bellekten silinmesi.
free(satirDizi);
free(satir);
free(dosyaPointer);
}

```

Main Fonksiyon Ekran Görüntüsü-2



Ad	Değiştirme tarihi	Tür	Boyut
bin	16.05.2022 21:46	Dosya klasörü	
doc	16.05.2022 19:49	Dosya klasörü	
src	16.05.2022 19:50	Dosya klasörü	
makefile	16.05.2022 20:16	Dosya	1 KB

Program Klasör Yapısı

```
C:\Users\asus\Desktop\B191200001-AleynaDikal-1B\C-LuhnAlgoritmasi-KaynakKod>mingw32-make -f makefile
gcc -o ./bin/sonuc ./src/program.c
./bin/sonuc
1234 1453 1323 4355 - gecersiz
4548 4746 6285 2625 - gecersiz
4345 2722 3233 4748 - gecersiz
8567 5746 2823 3837 - gecerli
8270 7321 4641 2200 - gecersiz
5130 5503 2197 4380 - gecersiz
9800 5869 5224 5512 - gecersiz
8363 1569 6952 7393 - gecerli
2946 9476 1668 3908 - gecersiz
2662 3403 9748 9936 - gecerli
1951 9198 8156 1979 - gecerli

C:\Users\asus\Desktop\B191200001-AleynaDikal-1B\C-LuhnAlgoritmasi-KaynakKod>
```

Örnek Ekran Çıktısı

KAYNAKÇA

- [1] <https://tr.nesrakonk.ru/luhn-algorithm/>
- [2] <https://tahayigitmelek.com/2021/05/07/luhn-algoritmasi/>
- [3] <https://www.matematiksel.org/luhn-algoritmasi-ile-kredi-kartiniz-nasil-dogrulanir/#:~:text=Luhn%20Algoritmas%C4%B1%2C%20k%C3%B6t%C3%BC%20niyetli%20sald%C4%B1r%C4%B1lara,olan%20mod%C3%BCler%20aritmetik%20%C3%BCzerine%20kuruludur.>
- [4] <https://www.coursehero.com/file/pqasb6v/Strengths-and-weaknesses-edit-The-Luhn-algorithm-will-detect-any-single-digit/>
- [5] <https://www.quora.com/What-are-the-disadvantages-of-using-Luhn-s-algorithm-for-credit-card-numbers>
- [6] <https://www.geeksforgeeks.org/luhn-algorithm/>
- [7] <https://www.yusufsezer.com.tr/c-dosya-islemleri/>
- [8] <https://www.muratoksuzer.com/2019/07/c-programlama-ile-txt-dosyayi-satir-satir-okuma.html>
- [9] <http://www.baskent.edu.tr/~tkaracay/etudio/ders/prg/c/string.htm>
- [10] https://www.tutorialspoint.com/c_standard_library/c_function_free.htm