T.C.
YILDIZ TEKNİK ÜNİVERSİTESİ
ELEKTRİK-ELEKTRONİK FAKÜLTESİ
ELEKTRONİK VE HABERLEŞME MÜHENDİSLİĞİ BÖLÜMÜ

PROGRAMLAMA DİLLERİ PROJE RAPORU

ŞİFRELEME TEKNİKLERİ

Ders Koordinatörü : Dr.Öğr.Üyesi Hafiza İrem TÜRKMEN ÇİLİNGİR

Emre Kaan Özkan 18014601
Uğur Mestçi 19014602
Uğur Kelleci 19014606

İstanbul, 2020

BÖLÜM 1

Şifreleme Teknikleri

1. Giriş

Eski çağlardan beri devletler, insanlar her zaman bilgileri, yazıları önemli olan yazışmaları şifrelemekle uğraşmış ve kendilerince şifreleme yöntemleri geliştirmişlerdir. Günümüzde ise artık neredeyse insanlar bilgilerini, verilerini, dosyalarını bilgisayarlar aracılığıyla saklıyorlar ve yazılım ile şifreleme yöntemleri büyük rağbet görmektedirler. Bundan dolayı artık günümüzde yeni şifreleme algoritmalarına ihtiyaç duyulur ve bunun için algoritmalar geliştirilir. Doğru kullanıldığında kırılması neredeyse imkansız olan şifreleme teknolojisi, elimizdeki bilgileri kötü aktörlerden ve servis sağlayıcılarından korumamız için günümüzde sahip olduğumuz en önemli araçlardan biri. Bu raporda şifreleme algoritmasıyla C dilinde yazılmış iki program örneği gözlemlenmiştir[1].

2. Görev 1 Metin Şifreleme ve Deşifreleme Yapan Program

Birinci görev üç programdan oluşmaktadır. İlk kısım şifreli matris oluşturma programıdır, ikinci program matris ile metin şifreleme işlevini görür.
Son programda ise metin şifresi matris kullanılarak çözülme işlemi yapılır. Bu programlar sırayla ele alınacaktır.

2.1 Matris Oluşturma Programında Kullanılan Kütüphanler

Matris oluşturmak için yazdığımız matrisOlustur.c programını incelemeye kütüphanelerden başlayabiliriz.

```
//
// Created by KAAN-UGUR-UGUR on 30.04.2020.
//
#include <stdio.h>                                  //Standart Giris Cikis Kutuphanesi Tanimlanmasi
#include <stdlib.h>                                 //Dinamik bellek islemleri icin kullanilan kutuphane
#include "time.h"                                   //Random islemler icin kullanilan kutuphane
```

Stdio.h : Standart fonksiyonları içeren kütüphanedir.

Stdlib.h : Dinamik bellek işlemlerini yapmak için kullandığımız kütüphanedir.

Time.h : Random sayılar üretmek için kullandığımız kütüphanedir. Bilgisayarın içindeki saati baz alarak hesaplamalar yapar.

2.2 Matris Oluşturma Programı Main Fonksiyonu

Matris oluşturma C programında diğer programlarda olduğu gibi ilk olarak main fonksiyonu çalışır. Main fonksiyonunda yaptığımız işlemler sırasıyla şöyledir :

- İlk kullanıcıdan matrisin şifreleneceği dosyayı uzantısı şeklinde istenmiştir.
- Sırasıyla satır ve sütun değerleri kullanıcıdan istenmiştir.
- Main fonksiyonunun içindeki matrise yerleştir, matrisi karıştır, matrisi dosyaya yaz fonksiyonları sayesinde şifrelenmiş matrisi içeren dosyamız oluşur.

```c
int main()
{
    int satir,sutun;                                                    //Satir ve sutun bilgisinin saklandigi degiskenler
    char dosya_ismi[MAX_DOSYA_ADI_UZUNLUGU];                            //(Buffer) Matrisin yazilacagi dosya adi icin
    char **sifre_matrisi;                                               //Sifrenin tutuldugu matris

    printf("Matrisin yazilacagi dosya ismini uzantisi ile birlikte veriniz : \n");   //Matrisin yazildigi dosyanin isminin istendigi kisim
    scanf("%s",dosya_ismi);

    printf("Matrisin satir ve sutun sayisini ayni sira ile veriniz (min 54 hucre) :\n ");   //Matrisin boyutunun istendigi kisim
    scanf("%d %d",&satir,&sutun);

    if(satir*sutun <54 || satir*sutun>81)                               //Boyutlarin yanlis girilmesi durumunda hata veriliyor
    {
        printf("Yanlis deger girdiniz");
        exit(0);
    }


    sifre_matrisi = yer_ac(satir,sutun);                                //Matris icin dinamik bellek ayrilmasi

    matrise_yerlestir(sifre_matrisi,satir,sutun);                       //Semboller matrise yerlestiriliyor.

    matris_karistir(sifre_matrisi,satir,sutun);                         //Matris karistiriliyor.

    matrisi_dosyaya_yaz(sifre_matrisi,satir,sutun,dosya_ismi);          //Matris dosyaya yazilir.

    printf("Sifre matrisi dosyasi olusturuldu..");                      //Islemin tamamlandigi bildiriliyor.

    free_memory(sifre_matrisi,satir);                                   //Dinamik bellek free ediliyor.

}
```

Figür 2.1 MatrisOlustur.c Main Fonksiyonu

```
Matrisin yazilacagi dosya ismini uzantisi ile birlikte veriniz :
sifreMatrisi.txt
Matrisin satir ve sutun sayisini ayni sira ile veriniz (min 54 hucre) :
7 8
Sifre matrisi dosyasi olusturuldu..
--------------------------------
Process exited after 4.683 seconds with return value 1
Press any key to continue . . .
```

```
satir= 7 sutun= 8
vCDMlEhcdnpxgTObwQR yBKIGVmzokuLPfZXt SYrNsjaFUqiJAWeH.
```

Figür 2.2 MatrisOlustur.c Output ve sifreMatrisi.txt

2.3 Matris Oluşturma Programı Fonksiyonları

Bu kısımdaki **ilk fonksiyonumuz**, yer açma[2] fonksiyonumuzdur. Bu fonksiyon şifre matrisi için dinamik bellek yönetimiyle yer açmamızı sağlar. Matrisi tekrar döndürerek main de tanımladığımız şifre matrisine eşitledik. Bu sayede matris boyutunca yer açma işlemi gerçekleştirildi. Yer açmamızın nedeni satır ve sutunu kullanıcının belirlediği matris boyutunca yere ihtiyacımız olmasıdır. C dilinde matris[n] gibi içinde değişken bulunan diziler tanımlanamaz.

```
char ** yer_ac(int satir,int sutun)                                    //Matris icin dinamik bellek ayirmayi saglayan fonksiyon
{
    int i;
    char ** sifre_matrisi;
    sifre_matrisi = (char **) calloc (satir,sizeof(char*));
    for (i = 0; i<satir;i++)
    {
        sifre_matrisi[i]=(char*) calloc (sutun,sizeof(char));          //char matrisi oldugu icin matris boyutlarina gore
                                                                       //  sizeof(char) kadar alan ayriliyor.
    }
    return sifre_matrisi;
```

Figür 2.3 Yer Açmak için Kullanılan Fonksiyon

İkinci fonksiyonumuz, matrise yerleştir fonksiyonumuzdur bu fonksiyonda ASCII kodlarını sayac ile kontrol ederek alfabedeki büyük ve küçük bütün harflerimizi, boşluk ve nokta karakterlerini matrisimize sırasıyla yerleştirilmiştir. Boş kalan yerler "NULL" karakter ile doldurulmuştur.

```
void matrise_yerlestir (char** sifre_matrisi, int satir, int sutun)          //Sembolleri matrise yerlestiren fonksiyon
{   int i,j;
    int sayac =65;
    for (i = 0; i<satir; i++)                                                //Matris boyutlarina gore yerlestirme yapilir.
    {
        for (j = 0; j < sutun; j++)
        {
            sifre_matrisi[i][j] = sayac;                                     //Sayacin degerine gore ASCII degerleri matrise atanir.
            sayac++;
            if (sayac == 91)
            {
                sayac = 97;
            }
            else if (sayac == 123)                                           //Nokta,bosluk degerleri icin kontrol saglanan bloklar
            {
                sayac = 46;
            }
            else if (sayac == 47)
            {
                sayac = 32;
            }
            else if(sayac >= 0 && sayac < 32 || sayac > 32 && sayac < 65)    //Diger degerler icin NULL atamasi yapilan blok
            {
                sayac = 0;
            }
        }

    }
}
```

Figür 2.4 Harf, Nokta ve Boşluğu Matrise Yerleştiren Fonksiyon

Üçüncü fonksiyonumuz, matrisin karıştırmamızı sağlayan fonksiyonumuzdur. Burada time kütüphanesinde yer alan srand[3] fonksiyonunu kullanarak matrisi karıştırmak için rastegele değerler atadık. Ve matrisimizi karıştırdık.

```
void matris_karistir (char** sifre_matrisi, int satir, int sutun)        //Matrisin karistirilmasini saglayan fonksiyon
{
    int i,j;
    srand(time(NULL));                                                   //Zamana göre rand degerler uretmek icin cagirdigimiz fonksiyon
    for (i = 0; i<satir; i++)
    {
        for (j = 0; j < sutun; j++)
        {
            int a = rand()%(satir-1);                                    //Karistirma icin random degerler uretilmesi.
            int b = rand()%(sutun-1);
            int temp= sifre_matrisi[i][j];
            sifre_matrisi[i][j] =sifre_matrisi[a][b];
            sifre_matrisi[a][b] = temp;

        }
    }
}
```

Figür 2.5 Matrisi Karıştırmak İçin Kullanılan Fonksiyon

Matris oluşturma programımızdaki **dördüncü fonksiyonumuz**, oluşturulan matrisi kullanıcıdan aldığımız dosyaya yazmamızı sağlayan fonksiyondur. Fonksiyon dosyayı oluşturup içine öncelikle şifre matrisinin boyutunu sonrada kendisini yerleştirmektedir.Bu fonksiyonda fputc ve fprintf gibi fonksiyonlar ile text dosyasına yazma işlemi gerçekleştirilmektedir.

```
void matrisi_dosyaya_yaz(char** sifre_matrisi, int satir, int sutun, char* dosya_ismi)          //Satir-sutun bilgisi ve matrisi dosyaya yazan fonksiyon
{
    FILE* sifre_dosyasi;                                                                        //Dosya ile programin iletisim kurmasinn saglamak icin tanimlandi
    int i,j;                                                                                    //Dongu degiskenleri
    sifre_dosyasi = fopen(dosya_ismi,"w");                                                      //Dosya yazma modunda aciliyor.
    if (sifre_dosyasi == NULL)                                                                  //Dosya acilamazsa hata dondurerek program kapatilir.
    {
        printf("error!! \n");
        exit(0);
    }
    fprintf(sifre_dosyasi,"satir= %d sutun= %d",satir,sutun);                                   //Satir ve sutun bilgisi yaziliyor.
    fprintf(sifre_dosyasi," ");
    for(i = 0; i<satir;i++)
    {
        for(j=0; j<sutun;j++)
        {

            fputc(sifre_matrisi[i][j],sifre_dosyasi);                                           //Matristeki bilgili yazmak icin kullanilan fonksiyon

        }
    }
    fclose(sifre_dosyasi);                                                                      //Dosyalari kapatmak icin kullanilan fonksiyon
}
```

Figür 2.6 Matrisi Text Dosyasına Yazan Fonksiyon

Bir diğer fonksiyonumuz, matris için ayrılan belleğin free edilmesini sağlar[2].

```
void free_memory(char** sifre_matrisi, int satir)          //Matris icin ayrilan bellegi free etmek icin kullanilan fonksiyon
{   int i;
    for (i=0;i<satir;i++)
    {
        free(sifre_matrisi[i]);                            //Matrise ayrilan alan kadar dongu icerisinde free fonksiyonu ile alan bosa cikartilir.
    }
    free(sifre_matrisi);
}
```

Figür 2.7 Matris için Açılan Dinamik Belleği Free Eden Fonksiyon

Son olarak matrisi kontrol etmek için kullandığımız matrisi ekrana yazdıran fonksiyon ile programımızın fonksiyonlarını bitirmiş bulunmaktayız.

```
void matrisi_ekrana_yazdir(char** sifre_matrisi, int satir, int sutun)          //Matrisi ekrana yazdiran fonksiyon
{   int i,j;
    for(i = 0; i<satir;i++)
    {
        for(j=0; j<sutun;j++)
        {
            printf("%c ",sifre_matrisi[i][j]);

        }
        printf("\n");
    }
```

Figür 2.7 Matrisi Ekrana Yazdıran Fonksiyon

ÖNEMLİ NOT : Matris oluştur fonksiyonunda 7 8 gibi bir takım boyut bilgileri girildiği takdirde UTF-8'den dolayı çince txt dosyaları oluşabilmektedir. Bu dosyaları görüntülerken windows text görüntüleyicisi yerine WORDPAD, VSCODE gibi programlar kullanırsanız düzgün bir şekilde yazma işleminin yapıldığı gözlemlenecektir.

2.4 Metin Şifreleme Programı Kullanılan Kütüphaneler

İkinci programımız olan metinSifrele.c C programında kullandığımız kütüphaneler figür 2.8'de verilmiştir.

```
//
// Created by KAAN-UGUR-UGUR on 30.04.2020.
//
#include<stdio.h>                    //Standart Giris Cikis Kutuphanesi Tanimlanmasi
#include<stdlib.h>                   //Dinamik bellek islemleri icin kullanilan kutuphane
```

Figür 2.8 Matrisi Ekrana Yazdıran Fonksiyon

Stdio.h : Standart fonksiyonları içeren kütüphanedir.

Stdlib.h : Dinamik bellek işlemlerini yapmak için kullandığımız kütüphanedir.

2.5 Metin Şifreleme Programı Main Fonksiyonu

Metin şifrelemek için yazdığımız programın main fonksiyonunu inceleyecek olursak :

- İlk olarak kullanıcıdan şifreli matrisin yazılı olduğu dosyayı uzantısıyla birlikte istenmiştir.
- Şifre matrisinin bulunduğu dosyadan okunan satır sutun bilgisine göre M için yer açılır bunun nedenini yukarıda açıklamıştık. M için dinamik yer açılması gereklidir.
- Text'ten okunan şifre dizide tutulup "M "matrisine yerleştirilir.
- Ardından kullanıcıdan şifrelenecek olan metin uzantısıyla birlikte istenmiştir.
- Şifreli metinin yerleştirileceği dosyanın ismi uzantısıyla birlikte istenmiştir.
- Son olarak metin_oku_sifrele_yaz fonksiyonumuz sayesinde şifreli metin oluşturulmuştur olup dosyaya yazılmıştır.

```
int main()
{   int satir_sutun[2];                                          //Satir ve sutun bilgisinin saklandigi dizi
    char sifre_matrisi_dosya_adi[MAX_DOSYA_ADI_UZUNLUGU];        //(Buffer) Matrisin alindigi dosya adi icin
    char sifrelenecek_dosya_adi[MAX_DOSYA_ADI_UZUNLUGU];         //(Buffer) Metinin alindigi dosya adi icin
    char sifrelenmis_dosya_adi[MAX_DOSYA_ADI_UZUNLUGU];          //(Buffer) Sifrelenecek metinin alindigi  dosya adi icin
    char chDizi[1000];                                           //Sifre matrisinin ilk tutuldugu dizi
    char **M;                                                    //Sifre matrisinin tutuldugu, sifreleme islemlerinde kullanilan 2D dizi

    printf("Matrisin yazildigi dosya ismini uzantisi ile birlikte veriniz : \n");   //Sifre matrisinin istendigi kisim
    scanf("%s",sifre_matrisi_dosya_adi);

    sifreyi_oku(chDizi,sifre_matrisi_dosya_adi,satir_sutun);     //Sifre ilk etapta matristen okunup bir dizide tutuluyor.
    M = yer_ac_matris(satir_sutun);                              //Gelen satir ve sutun bilgisine gore M' matrisine dinamik bellek ayrildi.
    sifreyi_matrise_koy(M,chDizi,satir_sutun);                   //Sifre yerlestirildi.

    printf("Sifrelenecek metni iceren dosya ismini uzantisi ile birlikte veriniz : \n");   //Sifrelenecek metin istendi.
    scanf("%s",sifrelenecek_dosya_adi);

    printf("Sifrelenmis metni icerecek dosya ismini uzantisi ile birlikte veriniz: \n");   //Sifre dosyasi isteniyor
    scanf("%s",sifrelenmis_dosya_adi);

    metin_oku_sifrele_yaz(sifrelenecek_dosya_adi,M,satir_sutun,sifrelenmis_dosya_adi);     //Alinan bilgiler dogrultusunda metin okunup sifrelenip yaziliyor.

    printf("Sifreleme tamamlanmistir");

    free_memory_ch_matrix(M,satir_sutun);                        //Matrise ayrilan dinamik bellek free edildi.

}
```

Figür 2.9 Matris Şifreleme İşlemi Yapan Programın Main Fonksiyonu

```
Matrisin yazildigi dosya ismini uzantisi ile birlikte veriniz :
sifreMatrisi.txt
Sifrelenecek metni iceren dosya ismini uzantisi ile birlikte veriniz :
littlePrince.txt
Sifrelenmis metni icerecek dosya ismini uzantisi ile birlikte veriniz:
littlePrinceEncrypted.txt
Sifreleme tamamlanmistir
--------------------------------
```

```
87 54 43 26 74 23 11 26 54 74 33 74 23 71 83 74 83 63 47 74 41 26 71 85 83 74 51 15 13 74 33 74 83 71 23 74 71 74 5
6 74 56 51 74 53 51 64 26 74 74 71 54 13 74 56 11 26 41 74 83 15 26 26 55 74 56 11 85 51 84 62 11 74 56 11 26 74 83
```

Figür 2.10 MetinSifrele.c Output ve LittlePrinceEncrypted.txt

Çıktıdan gözlemlediğimiz üzer matrise göre metinimiz şifrelenip txt dosyasına kaydedilmiştir. Örneği “O” harfi 6.satır 7.sütundadır şifre (sütunNo+1)*10+(satır No+1)[3] işlemine göre 87 olarak belirlemiştir.

2.6 Metin Şifreleme Programı Fonksiyonları

Metni şifreleme kısımdaki ilk fonksiyonumuz şifreyi oku fonksiyonumuzdur. Öncelikle boyut bilgisi text’ten okunur.Sonra matris okunup bir diziye yerleştirilir.

```
void sifreyi_oku(char chDizi[],char* dosya_adi,int satir_sutun_dizisi[])                    //Dosyadan Matrisi ve boyutlarini okuyan fonksiyon
{
    FILE *sifre_matris_dosyasi;                                                             //Dosya ile programin iletisim kurmasini saglamak icin tanimlandi.
    int satir_degeri, sutun_degeri;                                                         //Okunan satir ve sutun degerlerini tutmak icin kullanildi.
    int n=0;                                                                                //Diziye aktarmak icin kullanildi.
    sifre_matris_dosyasi = fopen(dosya_adi, "r");                                           //Dosyayi okuma modunda acmak icin kullanilan fonksiyon.
    if(sifre_matris_dosyasi == NULL)                                                        //Dosya okunamaz ise hata dondurup programi sonlandirir.
    {
        printf("Error opening file\n");
        exit(1);
    }
    if( fscanf(sifre_matris_dosyasi, "satir=%d sutun=%d",&satir_degeri,&sutun_degeri) != EOF )   //Dosyanin sonunda degilse satir ve sutun degerleri okunur(Biz basta kaydetmistik).
    {
        satir_sutun_dizisi[0]=satir_degeri;                                                 //Okunan degerleri diziye kaydediyoruz Main'de M matrisine yer acmak icin
        satir_sutun_dizisi[1] = sutun_degeri;
        while (!feof(sifre_matris_dosyasi))
        {
            chDizi[n]=getc(sifre_matris_dosyasi);                                           //Matris okunuyor ve dizide tutuluyor
            n++;
        }
    }
    fclose(sifre_matris_dosyasi);
}
```

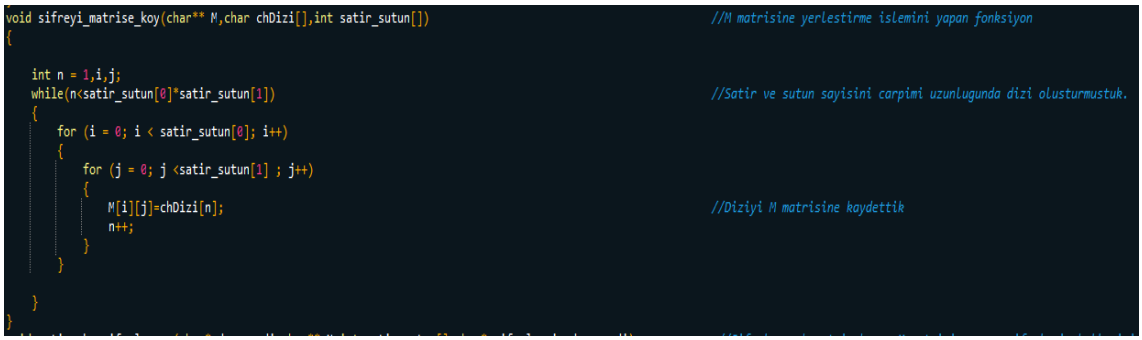
Figür 2.11 Text’ten Şifreyi Okuyan Fonksiyon

İkinci fonksiyonumuz daha önceki programımızda kullandığımız yer açma fonksiyonudur.Buradaki görevi “M” matrisine dinamik bellek ayırmaktır.

```
char ** yer_ac_matris(int satir_sutun[])                                    //Matris icin dinamik bellek ayirmayi saglayan fonksiyon
{
    int i;
    char **sifre_matrisi;
    sifre_matrisi = (char **) calloc(satir_sutun[0], sizeof(char *));
    for (i = 0; i < satir_sutun[0]; i++)
    {
        sifre_matrisi[i] = (char *) calloc(satir_sutun[1], sizeof(char));       //Char matrisi oldugu icin matris boyutlarina gore sizeof(char) kadar alan ayriliyor.
    }
    return sifre_matrisi;
}
```

Figür 2.12 Text’ten Şifreyi Okuyan Fonksiyon

Sifreyi_matrise_koy fonksiyonu, şifreyi oku fonksiyonunda bir dizide tuttuğumuz değerleri M matrisine yerleştirmemizi sağlıyor.



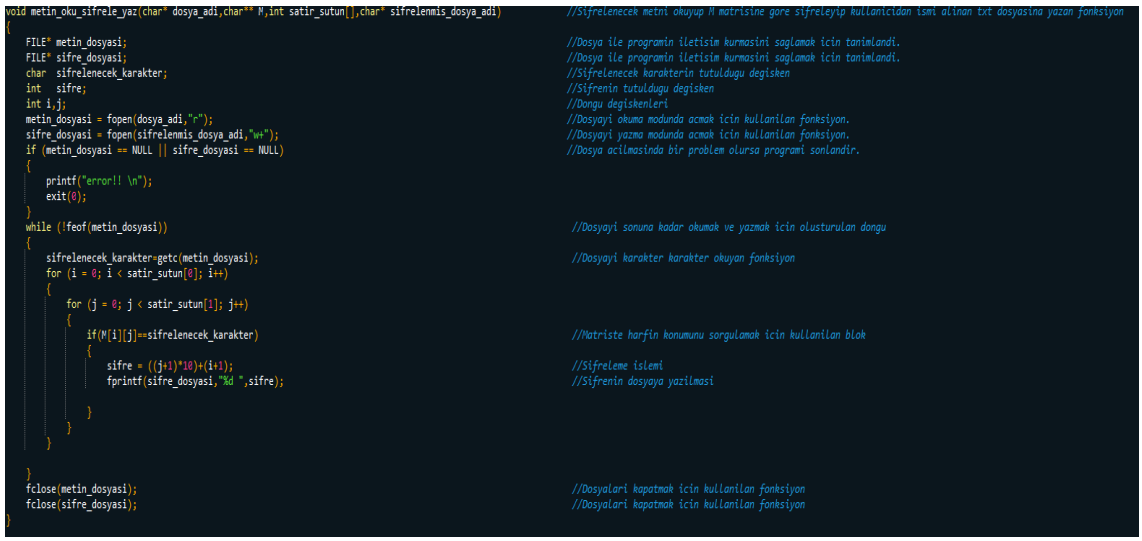
```
void sifreyi_matrise_koy(char** M,char chDizi[],int satir_sutun[])                    //M matrisine yerlestirme islemini yapan fonksiyon
{

    int n = 1,i,j;
    while(n<satir_sutun[0]*satir_sutun[1])                                            //Satir ve sutun sayisini carpimi uzunlugunda dizi olusturmustuk.
    {
        for (i = 0; i < satir_sutun[0]; i++)
        {
            for (j = 0; j <satir_sutun[1] ; j++)
            {
                M[i][j]=chDizi[n];                                                    //Diziyi M matrisine kaydettik
                n++;
            }
        }

    }
}
```

Figür 2.13 Text'ten Okunan Şifreyi M Matrisine Yerleştiren Fonksiyon

Dördüncü fonksiyonumuz, bu programın en önemli fonksiyonu olup şifreleme işlemini ve oluşan bu şifreyi dosyaya yazma işlemini yapmaktadır.
> İlk olarak şifrelenecek olan metni karakter karakter okuyor. Bu karakterlerin bulunduğu matristeki satır ve sutun numaralarını bulmuştur .
> Fonksiyon karakterleri (sütunNo+1)*10+(satır No+1)[3] işlemine göre şifrelemiştir.
> Ve son olarak bu şifreli metin kullanıcıdan alınan dosya ismine göre dosyaya yazılmıştır.
Metin karakter karakter okunmaktadır. Burada önemli olan nokta metin okunduğu an şifrelenmiştir. Bu sebepten dolayı ayrı bir ortamda onu tutmamız gerekmemiştir. Ve tekrar memory ayırmamız gerekmemiştir.



```
void metin_oku_sifrele_yaz(char* dosya_adi,char** M,int satir_sutun[],char* sifrelenmis_dosya_adi)   //Sifrelenecek metni okuyup M matrisine gore sifreleyip kullanicidan ismi alinan txt dosyasina yazan fonksiyon
{
    FILE* metin_dosyasi;                                                                  //Dosya ile programin iletisim kurmasini saglamak icin tanimlandi.
    FILE* sifre_dosyasi;                                                                  //Dosya ile programin iletisim kurmasini saglamak icin tanimlandi.
    char  sifrelenecek_karakter;                                                          //Sifrelenecek karakterin tutuldugu degisken
    int   sifre;                                                                          //Sifrenin tutuldugu degisken
    int i,j;                                                                              //Dongu degiskenleri
    metin_dosyasi = fopen(dosya_adi,"r");                                                 //Dosyayi okuma modunda acmak icin kullanilan fonksiyon.
    sifre_dosyasi = fopen(sifrelenmis_dosya_adi,"w+");                                    //Dosyayi yazma modunda acmak icin kullanilan fonksiyon.
    if (metin_dosyasi == NULL || sifre_dosyasi == NULL)                                   //Dosya acilmasinda bir problem olursa programi sonlandir.
    {
        printf("error!! \n");
        exit(0);
    }
    while (!feof(metin_dosyasi))                                                           //Dosyayi sonuna kadar okumak ve yazmak icin olusturulan dongu
    {
        sifrelenecek_karakter=getc(metin_dosyasi);                                         //Dosyayi karakter karakter okuyan fonksiyon
        for (i = 0; i < satir_sutun[0]; i++)
        {
            for (j = 0; j < satir_sutun[1]; j++)
            {
                if(M[i][j]==sifrelenecek_karakter)                                         //Matriste harfin konumunu sorgulamak icin kullanilan blok
                {
                    sifre = ((j+1)*10)+(i+1);                                              //Sifreleme islemi
                    fprintf(sifre_dosyasi,"%d ",sifre);                                    //Sifrenin dosyaya yazilmasi

                }
            }
        }

    }
    fclose(metin_dosyasi);                                                                 //Dosyalari kapatmak icin kullanilan fonksiyon
    fclose(sifre_dosyasi);                                                                 //Dosyalari kapatmak icin kullanilan fonksiyon
}
```

Figür 2.14 Text'ten Metni Okuyup Şifreleyen Fonksiyon

Son fonksiyonumuz, M matrisi için ayrılan belleğin free edilmesini sağlar.

```
void free_memory(char** sifre_matrisi, int satir)          //Matris icin ayrilan bellegi free etmek icin kullanilan fonksiyon
{   int i;
    for (i=0;i<satir;i++)
    {
        free(sifre_matrisi[i]);                            //Matrise ayrilan alan kadar dongu icerisinde free fonksiyonu ile alan bosa cikartilir.
    }
    free(sifre_matrisi);
}
```

Figür 2.15 M Matrisi için Açılan Alanı Free Eden Fonksiyon

2.7 Şifre Çözme Programı Kütüphaneleri

Son programımız olan metinSifreCozme.c C programında kullandığımız kütüphaneler figür 2.16'da verilmiştir.

```
//
// Created by KAAN-UGUR-UGUR on 30.04.2020.
//
#include<stdio.h>                                  //Standart Giris Cikis Kutuphanesi Tanimlanmasi
#include<stdlib.h>                                 //Dinamik bellek islemleri icin kullanilan kutuphane
```

Figür 2.16 Şifre Çözme Programında Kullanılan Kütüphaneler

Stdio.h : Standart fonksiyonları içeren kütüphanedir.

Stdlib.h : Dinamik bellek işlemlerini yapmak için kullandığımız kütüphanedir.

2.8 Şifre Çözme Programı Main Fonksiyonu

Şifrelenmiş metini çözem C programımızın Main fonksiyonunu inceleyecek olursak:

- İlk olarak kullanıcının bizden ilk programda oluşturduğumuz şifreli matris dosyasını uzantısıyla birlikte istenmiştir. (Örnek olarak "LittlePrince.txt"[3] kullanılmıştır.)
- M matrisi için yer açılmıştır.
- Şifreyi okunur. Sonrasında şifre M matrisine yerleştirilir.
- Son olarak şifreli metni okuyup çözen ve ardından kullanıcıdan alınan bilgiler doğrultusunda oluşturulan dosyaya yazan fonksiyonumuz sayesinde metnin şifresi çözülmüş ve dosya oluşturulmuştur. Burada karakterler tek tek okunup, M matrisiyle karşılaştırılıp, çözülüp tek tek yazılmışlardır.

- Çıktılardanda anlaşılacağı üzere şifremiz çözülerek tekrar text'e kaydedilmiştir.

```c
int main()
{   int satir_sutun[2];                                                  //Satir ve sutun bilgisinin saklandigi dizi
    char sifre_matrisi_dosya_adi[MAX_DOSYA_ADI_UZUNLUGU];                //(Buffer) Matrisin alindigi dosya adi icin
    char sifrelenmis_metin_dosya_adi[MAX_DOSYA_ADI_UZUNLUGU];            //(Buffer) Sifreli metnin alindigi dosya adi icin
    char cozulmus_metin_dosya_adi[MAX_DOSYA_ADI_UZUNLUGU];               //(Buffer) Cozulmus metnin yazilacagi dosya adi icin
    char chDizi[1000];                                                   //Sifre matrisinin ilk tutuldugu dizi
    char **M;                                                            //Sifre matrisinin tutuldugu, sifreleme islemlerinde kullanilan 2D dizi

    printf("Matrisin yazildigi dosya ismini uzantisi ile birlikte veriniz : \n");   //Sifre matrisinin dosya adinin istendigi kisim
    scanf("%s",sifre_matrisi_dosya_adi);

    sifreyi_oku(chDizi,sifre_matrisi_dosya_adi,satir_sutun);             //Sifre ilk etapta matristen okunup bir dizide tutuluyor.
    M = yer_ac_matris(satir_sutun);                                      //Gelen satir ve sutun bilgisine gore M' matrisine dinamik bellek ayrildi.
    sifreyi_matrise_koy(M,chDizi,satir_sutun);                           //Sifre M'e yerlestirildi.

    printf("Sifrelenmis metni iceren dosya ismini uzantisi ile birlikte veriniz : \n");  //Sifrelenmis metin dosya istendi.
    scanf("%s",sifrelenmis_metin_dosya_adi);

    printf("\nSifresi cozulmus metni icerecek dosya ismini uzantisi ile birlikte veriniz: \n");  //Cozulmus metnin yazilacagi dosya adi istendi.
    scanf("%s",cozulmus_metin_dosya_adi);

    sifreli_metni_oku_coz_yaz(sifrelenmis_metin_dosya_adi,M,satir_sutun,cozulmus_metin_dosya_adi);  //Alinan bilgiler dogrultusunda sifre okunup cozulup yaziliyor.

    free_memory_ch_matrix(M,satir_sutun);                                //Matrise ayrilan dinamik bellek free edildi.

}
```

Figür 2.17 Şifre Çözme Programı Main Fonksiyonu

```
Matrisin yazildigi dosya ismini uzantisi ile birlikte veriniz :
sifreMatrisi.txt
Sifrelenmis metni iceren dosya ismini uzantisi ile birlikte veriniz :
littlePrinceEncrypted.txt
Sifresi cozulmus metni icerecek dosya ismini uzantisi ile birlikte veriniz:
littlePrinceDecrypted.txt
Sifre cozme tamamlanmistir.
```

```
Once when I was six years old I saw a magnificent picture in a book called True Stories from Nature about the prime
```

Figür 2.18 Şifre Çözme Programı Output ve LittlePrinceDecrypted.txt

2.9 Şifre Çözme Programı Fonksiyonları

İlk fonksiyonumuz, daha önce şifreleme işlemini yaparken kullandığımız bir fonksiyon olup ilk kısımda oluşturduğumuz şifre matrisini içeren dosyasını okuyor ve oradaki değerleri bir dizide tutmamızı sağlıyor.

```c
void sifreyi_oku(char chDizi[],char* dosya_adi,int satir_sutun_dizisi[])     //Dosyadan Matrisi ve boyutlarini okuyan fonksiyon
{
    FILE *sifre_matris_dosyasi;                                              // Dosya ile programin iletisim kurmasinn saglamak icin tanimlandi.
    int satir_degeri, sutun_degeri;                                          //Okunan satir ve sutun degerlerini tutumak icin kullanildi.
    int n=0;                                                                 //Diziye aktarmak icin kullanildi.
    sifre_matris_dosyasi = fopen(dosya_adi, "r");                            //Dosyayi okuma modunda acmak icin kullanilan fonksiyon
    if(sifre_matris_dosyasi == NULL)                                         //Dosya okunamaz ise hata dondurup programi sonlandirir.
    {
        printf("Error opening file\n");
        exit(1);
    }
    if( fscanf(sifre_matris_dosyasi, "satir=%d sutun=%d",&satir_degeri,&sutun_degeri) != EOF )   //Dosyanin sonunda degilse satir ve sutun degerleri okunur
    {
        satir_sutun_dizisi[0]=satir_degeri;                                  //Okunan degerleri diziye kaydediyoruz Main'de M matrisine yer acmak icin
        satir_sutun_dizisi[1] = sutun_degeri;
        while (!feof(sifre_matris_dosyasi))
        {
            chDizi[n]=getc(sifre_matris_dosyasi);                            //Matris okunuyor ve dizide tutuluyor
            n++;
        }
    }
    fclose(sifre_matris_dosyasi);
}
void sifreyi_matrise_koy(char** M,char chDizi[],int satir_sutun[])           //M matrisine yerlestirme islemini yapan fonksiyon
```

Figür 2.19 Text'ten Şifreyi Okuyan Fonksiyon

İkinci fonksiyonumuz, "M" matrisine dinamik bellek ayırmamızı sağlamak için yazılmıştır.

```
char ** yer_ac_matris(int satir_sutun[])                    //Matris icin dinamik bellek ayirmayi saglayan fonksiyon
{
    int i;
    char **sifre_matrisi;
    sifre_matrisi = (char **) calloc(satir_sutun[0], sizeof(char *));   //char matrisi oldugu icin matris boyutlarina gore sizeof(char) kadar alan ayriliyor.
    for (i = 0; i < satir_sutun[0]; i++) {
        sifre_matrisi[i] = (char *) calloc(satir_sutun[1], sizeof(char));
    }
    return sifre_matrisi;
}
```

Figür 2.20 Text'ten Şifreyi Okuyan Fonksiyon

Üçüncü fonksiyonumuz, ilk fonksiyonumuzda bir dizide tutulan değerleri oluşturduğumuz "M" matrisine yerleştirme işlemini yapmaktadır.

```
void sifreyi_matrise_koy(char** M,char chDizi[],int satir_sutun[])          //M matrisine yerlestirme islemini yapan fonksiyon
{
    int n = 1,i,j;
    while(n<satir_sutun[0]*satir_sutun[1])                                  //Satir ve sutun sayisini carpimi uzunlugunda dizi olusturmustuk.
    {
        for (i = 0; i < satir_sutun[0]; i++)
        {
            for (j = 0; j <satir_sutun[1] ; j++)
            {
                M[i][j]=chDizi[n];                                          //Diziyi M matrisine kaydettik
                n++;
            }
        }

    }
}
```

Figür 2.21 Okunan Şifreyi Matrise Yerleştiren Fonksiyon

Şifreli metni çözen programın bir diğer fonksiyonu , matris için ayrılan belleğin free edilmesini sağlar.

```
void free_memory(char** sifre_matrisi, int satir)       //Matris icin ayrilan bellegi free etmek icin kullanilan fonksiyon
{   int i;
    for (i=0;i<satir;i++)
    {
        free(sifre_matrisi[i]);                         //Matrise ayrilan alan kadar dongu icerisinde free fonksiyonu ile alan bosa cikartilir.
    }
    free(sifre_matrisi);
}
```

Figür 2.22 M Matrisini Free Eden Fonksiyon

Şifreli metni çözen programın son ve en önemli fonksiyonu sifreli_metni_oku_coz_yaz fonksiyonudur ki bu programın bel kemiğidir.

- İlk olarak şifreli metin C'nin dosya işlemleri fonksiyonları kullanılarak okunur.
- Tek tek okunan bu karakterler M matrisine yerleştirilen şifre matrisinin elemanlarıyla karşılaştıralarak tekrar çözülür.
- Çözülen her karakter kullanıcıdan ismi alınan text dosyasına tekrar yazılır.

```
void sifreli_metni_oku_coz_yaz(char* dosya_adi,char** M,int satir_sutun[],char* cozulmus_dosya_adi)    //Sifreyi okuyup, sifresini cozup, cozumu txt'ye kaydeden fonksiyon
{
    FILE* sifreli_dosya;                                                                               // Dosya ile programin iletisim kurmasinn saglamak icin tanimlandi
    FILE* cozulmus_dosya;                                                                              // Dosya ile programin iletisim kurmasinn saglamak icin tanimlandi
    int sifre;                                                                                         //Okunan sifreyi tutmak icin kullanilan degisken
    int i,j;                                                                                           //Dongu degiskenleri
    char cozulmus_karakter;                                                                            //Cozulen karakterleri tutmak icin kullanilan degisken

    sifreli_dosya = fopen(dosya_adi,"r");                                                              //Dosyayi okuma modunda acmak icin kullanilan fonksiyon.
    cozulmus_dosya = fopen(cozulmus_dosya_adi,"w+");                                                   //Dosyayi yazma modunda acmak icin kullanilan fonksiyon.
    if (sifreli_dosya == NULL || cozulmus_dosya == NULL)                                               //Dosya acilmasinda bir problem olursa programi sonlandir.
    {
        printf("error!! \n");
        exit(0);
    }
    while (!feof(sifreli_dosya))                                                                       //Dosyayi sonuna kadar okumak ve yazmak icin olusturulan dongu
    {
        fscanf(sifreli_dosya,"%d ",&sifre);                                                            //Dosyayi mesaj mesaj okuyan fonksiyon
        for (i = 0; i < satir_sutun[0]; i++)
        {
            for (j = 0; j < satir_sutun[1]; j++)
            {
                if(sifre == ((j+1)*10)+(i+1))                                                          //Sifrenin matristeki satir sutun degerini sorgulayan blok
                {   cozulmus_karakter = M[i][j];
                    fprintf(cozulmus_dosya,"%c",cozulmus_karakter);                                    //Cozulmus metni dosyaya yazin fonksiyon.
                }
            }
        }
    }
    fclose(sifreli_dosya);                                                                             //Dosyalari kapatmak icin kullanilan fonksiyon
    fclose(cozulmus_dosya);                                                                            //Dosyalari kapatmak icin kullanilan fonksiyon
}
```

Figür 2.23 Okunan Şifreyi Matrise Yerleştiren Fonksiyon

3. Görev 2 Matris Sezar Algoritması

Bu görevde verilen metni şifrelemek için bir şifreleme algoritması geliştirilmiştir. Daha sonrasında şifrelenen metnin deşifre edilip ekrana gösterilmesi sağlanmıştır. Bu işlemler genel olarak iki başlıkta ele alınmıştır. Öncelikle sistemin işleyişini anlatan **algoritma** bölümü ve bu algoritmanın hayata geçmesini sağlayan yazılımın tek tek fonksiyonlarının işlendiği **fonksiyonlar** bölümü maddeler halinde değerlendirilmiştir.

Şifreleme algoritması, Sezar[4] şifreleme sistemindeki harflerin alfabedeki sırasına göre belirli sayıda kaydırılması ve matris oluşturularak belirlenen şifreleme sisteminden esinlenilerek oluşturulmuştur.

3.1 Matris Sezar Algoritması

- -Şifreleme algoritması, kullanıcıdan stringin alınmasıyla başlar.

```
Sifrelenecek string'i giriniz :
elma
```

Figür 3.1 String İsteme

- Daha sonra stringteki her karakter için matriste kaç satır ve sütun mesafedeki (1-9 arasında) karakterin şifreleme için seçilmesine karar verilir. Böylelikle bir harf için belirlenen satır ve sütun sayısı kadar uzaklıkta iki karakter seçilmiş olur.

```
Sifrelenecek string'i giriniz :
elma
Sifre mesafelerini satir sutun icin belirtiniz (1-9 arasinda)
5
6
```

Figür 3.2 Şifrenin şifrelenecek harfe satır ve sütun olarak uzaklığı

- Matrisin boyutlarını belirlemek için kullanıcıya üç seçenek sunulur. Kullanıcı bunlardan birini seçer ve matris oluşur.

```
Sifrelenecek string'i giriniz :
elma
Sifre mesafelerini satir sutun icin belirtiniz (1-9 arasinda)
5
6
Matris boyutlarini seciniz :
 1.9-6  2.27-2  3.18-3 :
1
Sifre Matrisi
M r n O D T
z V X b E J
e d G C o B
l L v k f m
. y j K   u
Y s P t W h
N q w x A R
U Z a S g Q
I F H p i c
```

Figür 3.3 Şifre Matrisi Boyutları

- Karakterin matristeki yeri bulunur. Daha sonra girilen satır mesafesindeki karakter şifredeki ilk karakteri, girilen sütun mesafesindeki karakter de ikinci karakteri oluşturur.

```
Sifrelenecek string'i giriniz :
elma
Sifre mesafelerini satir sutun icin belirtiniz (1-9 arasinda)
5
6
Matris boyutlarini seciniz :
 1.9-6  2.27-2  3.18-3 :
1
Sifre Matrisi
M r n O D T
z V X b E J
e d G C o B
l L v k f m
. y j K   u
Y s P t W h
N q w x A R
U Z a S g Q
I F H p i c
Sifrelenmis metin :
UeIlcmva
```

Figür 3.4 Şifrelenmiş Metnin Ekrana Yazdırılması

- En sonunda şifrelenmiş metin deşifre edilerek ekrana yazdırılır.

```
Sifrelenecek string'i giriniz :
elma
Sifre mesafelerini satir sutun icin belirtiniz (1-9 arasinda)
5
6
Matris boyutlarini seciniz :
 1.9-6  2.27-2  3.18-3 :
1
Sifre Matrisi
M r n O D T
z V X b E J
e d G C o B
l L v k f m
. y j K   u
Y s P t W h
N q w x A R
U Z a S g Q
I F H p i c
Sifrelenmis metin :
UeIlcmva
Desifre Edilmis Metin :
elma
```

Figür 3.5 String'in Tekrar Deşifrelenip Ekrana Yazdırılması

Algoritmamızı kısaca özetleyecek olursak:

Nasıl ki Sezar şifreleme yaparken alfabedeki harflerin konumlarından belirli bir miktar öteleme yaparak bir şifreleme yöntemi oluşturduysa bizde kendi rasgele dizilmiş matrisimiz içinde kendi belirlediğimiz mesafeler kadar harfleri hareket ettirerek şifreleme işlemi yapıyoruz. Bir harf iki harf ile şifrelenmiştir. Birinci harfi satır mesafesi kadar uzakta ikinci harfi ise sütun mesafesi kadar uzakta alıp 4 harflik bir "elma" kelimesini 8 harf ile şifreleyip yollanmıştır.Örneğin "e" harfi 2.satır 0.sütundadır. 5 satır kaydırdığımız zaman 7. Satırda "U" harfi, 6 sütun kaydırdığımızda ise e harfi tekrar kendi üzerine gelerek şifrelenmiştir.

3.2 Sezar-Matris Algoritması Kütüphane Ve Fonksiyonları

Sezar matris algoritmasını incelemeye ilk olarak kütüphanelerden başlayabiliriz.

```
#include <stdio.h>                    //Standart Giris Cikis Kutuphanesi Tanimlanmasi
#include <stdlib.h>                   //Dinamik bellek islemleri icin kullanilan kutuphane
#include <time.h>                     //Random islemler icin kullanilan kutuphane
#include <string.h>                   //Bir takım string fonksiyonları barındıran kütüphane
#include <math.h>                     //Matematik islemleri icin kullanilan kutuphane
```

Figür 3.6 Sezar Matris Algoritması Kütüphaneleri

Stdio.h : Standart fonksiyonları içeren kütüphanedir.

Stdlib.h : Dinamik bellek işlemlerini yapmak için kullandığımız kütüphanedir.

Time.h : Random sayılar üretmek için kullandığımız kütüphanedir. Bilgisayarın içindeki saati baz alarak hesaplamalar yapar.

Math.h : abs,sqrt gibi özel matematik fonksiyonları içeren kütüphanedir.

String.h : Metin uzunluğunu alırken kullanacağımı strlen() fonksiyonu gibi string fonksiyonları içeren kütüphanedir.

Main fonksiyonumuz figür 3.7'de verilmiştir.

```
sifre_matrisi = yer_ac(satir,sutun);                    //Matris icin dinamik bellek ayrilmasi

matrise_yerlestir(sifre_matrisi,satir,sutun);             //Semboller matrise yerlestiriliyor.

matris_karistir(sifre_matrisi,satir,sutun);               //Matris karistiriliyor.

matrisi_ekrana_yazdir(sifre_matrisi,satir,sutun);

metin_uzunlugu_tutucu = sifrele(sifre_matrisi,satir,sutun,
sifrelenecek_metin,sifre_mesafe_satir,sifre_mesafe_sutun,sifreDizi,metin_uzunlugu_tutucu); //String matris kullanilarak sifreleniyor

desifre(sifre_matrisi,satir,sutun,sifre_mesafe_satir,sifre_mesafe_sutun,sifreDizi,metin_uzunlugu_tutucu); //Desifre islemini gerceklesitiren fonksiyon

free_memory(sifre_matrisi,satir);
```

Figür 3.7 Sezar Matris Algoritması Main Fonksiyonu

Main fonksiyonumuzu inceleyecek olursak :

- Öncelikle şifreleme işleminde kullanacağımız matris için yer açılmıştır.
- Karakterler matrise yerleştirilip karıştırılmıştır.
- Kontrolü sağlamak için matris ekrana yazdırılmıştır.
- Kullanıcıdan alınan satır ve sütun uzaklığı bilgisi sonucunda metin şifrelenmiştir.
- Şifreli metin ekrana yazdırıldıktan sonra deşifrelenerek program sonlandırılmıştır.

Bir diğer fonksiyonumuz dinamik bellek kullanımı için oluşturduğumuz yer_ac fonksiyonudur. Şifre matrisine yer açma işleminde kullanılır.

```
char ** yer_ac(int satir,int sutun)                    //Matris icin dinamik bellek ayirmayi saglayan fonksiyon
{
    int i;
    char **sifre_matrisi;
    sifre_matrisi = (char **) calloc(satir, sizeof(char *));
    for (i = 0; i < satir; i++) {
        sifre_matrisi[i] = (char *) calloc(sutun,sizeof(char));          //char matrisi oldugu icin matris boyutlarina gore sizeof(char) kadar alan ayriliyor.
    }
    return sifre_matrisi;
}
```

Figür 3.8 Yer Aç Fonksiyonu

Free fonksiyonu kullanılarak matris için ayrılan dinamik bellek boşaltılır.

```
char ** yer_ac(int satir,int sutun)                    //Matris icin dinamik bellek ayirmayi saglayan fonksiyon
{
    int i;
    char **sifre_matrisi;
    sifre_matrisi = (char **) calloc(satir, sizeof(char *));
    for (i = 0; i < satir; i++) {
        sifre_matrisi[i] = (char *) calloc(sutun,sizeof(char));          //char matrisi oldugu icin matris boyutlarina gore sizeof(char) kadar alan ayriliyor.
    }
    return sifre_matrisi;
}
```

Figür 3.9 Free Fonksiyonu

Matris boyutlarına göre karakterleri matrise yerleştirme işlemi gerçekleştiren void fonksiyon.

```
void matrise_yerlestir (char** sifre_matrisi, int satir, int sutun)          //Harf ve sembolleri matrise yerlestiren fonksiyon
{   int i,j;
    int sayac =65;
    for (i = 0; i<satir; i++)                          //Matris boyutlarina gore yerlestirme yapilir
    {
        for (j = 0; j < sutun; j++)
        {
            sifre_matrisi[i][j] = sayac;
            sayac++;
            if (sayac == 91)
            {
                sayac = 97;
            }
            else if (sayac == 123)
            {
                sayac = 46;
            }
            else if (sayac == 47)
            {
                sayac = 32;
            }
        }
    }
}
```

Figür 3.10 Matrise Yerleştirme İşlemi Yapan Fonksiyon

Rastgele değerler üretilerek matris karıştıran fonksiyon aşağıda verilmiştir.

```
void matris_karistir (char** sifre_matrisi, int satir, int sutun)       //Matrisin karistirilmasini saglayan fonksiyon
{
    int i,j;
    srand(time(NULL));                                  //Ayni degerler uretilmesin diye kullanilan fonksiyon
    for (i = 0; i<satir; i++)
    {
        for (j = 0; j < sutun; j++)
        {
            int a = rand()%(satir-1);                       //Karistirma icin random degerler uretilmesi.
            int b = rand()%(sutun-1);
            int temp= sifre_matrisi[i][j];
            sifre_matrisi[i][j] =sifre_matrisi[a][b];
            sifre_matrisi[a][b] = temp;
        }
    }
}
```

Figür 3.11 Matrisi Rastgele Karıştıran Fonksiyon<

Matrisi kontrol etmek için ekrana yazdırma ihtiyacı duyabiliriz. Bu sebepten dolayı figür 3.12'deki fonksiyon tanımlanmıştır.

```
void matrisi_ekrana_yazdir(char** sifre_matrisi, int satir, int sutun)          //Matrisi ekrana yazdiran fonksiyon
{   int i,j;
    printf("Sifre Matrisi \n");
    for(i = 0; i<satir;i++)
    {
        for(j=0; j<sutun;j++)
        {
            printf("%c ",sifre_matrisi[i][j]);
        }
        printf("\n");
    }
}
```

Figür 3.12 Matrisi Ekrana Yazdıran Fonksiyon

Sezar-Matris algoritmasının iki önemli fonksiyonu vardır. Bunlardan birincisi şifreleme işlemini gerçekleştiren **int sifrele()** fonksiyonudur. Bu fonksiyonun kısaca nasıl çalıştığını özetleyecek olursak :

- Fonksiyonda 3 adet döngü bulunmaktadır. While döngüsü ile stringin uzunluğu kadar işlem yapabilme yetisine sahip olmaktayız. For döngüler ise matrisin için kontrol etmektedir.
- Matristeki her bir karakter string'ten gelen karakterle karşılaştıralarak kullanıcıdan alınan satır mesafesi ve sütun mesafesine göre şifre harfleri elde edilir.
- Eğer son satır veya sütun'a gelirse tekrar başa dönerek arama işlemine devam edilir.

- Şifreleme işlemi bittiğinde şifrelenmiş metin bir dizide tutulur.
- En son bu dizi ekrana yazdırılır.

Sezar-Matris algoritmasının son fonksiyonu deşifreleme işlemi yapan fonksiyondur. Bu fonksiyon figür 3.13'te verilmiştir.

```
void desifre(char** sifrematrisi,int satir,int sutun,int satir_m,int sutun_m,int* sifreDizi,int metin_uzunlugu) //Desifreleme islemini yapan fonksiyon
{   int i,j;                                                                    //Dongu degiskenleri
    int a=0;                                                                    //Sayac
    int b;                                                                      //Sifrenin yerini bulmak icin matematiksel islemlerde kullanilan degiskenler
    int n = metin_uzunlugu ;                                                    //Sifre uzunlugu tutuluyor

    int desifre_edilecek_karakter;                                              //Karakter karakter desifreleme islemi yapilan fonksiyon
    int desifre_kar;
    int desifre_dizi[MAX_STRING_UZUNLUGU];
    int sayac = 0;

    while (a < n)                                                               //Stringi sonuna kadar okumak ve yazmak icin olusturulan dongu
    {
        desifre_edilecek_karakter = sifreDizi[a];                               //Stringi karakter karakter okumak icin kullanilan esitlik
        for (i = 0; i < satir; i++)
        {
            for (j = 0; j < sutun; j++)
            {
                if (sifrematrisi[i][j] == desifre_edilecek_karakter)            //Matriste harfin konumunu sorgulamak icin kullanilan blok
                {
                    if (i-satir_m >=0)                                          //Desifreleme icin matriste konum arama islemini yapan bloklar
                    {
                        b = i - satir_m % (satir);
                        desifre_kar = sifrematrisi[b][j];
                    }
                    else if (i - satir_m < 0)
                    {
                        b = abs(i - satir_m) ;
                        desifre_kar = sifrematrisi[satir - b][j];
                    }
                }
            }
        }
```

```
        a=a+2;
        desifre_dizi[sayac] = desifre_kar;                                      //Desifrelenmis metin diziye atiliyor.
        sayac++;
    }
    sayac=0;

    printf("\nDesifre Edilmis Metin : \n");

    if (n >1)
        n = n/2;
    while (sayac< n)
    {
        printf("%c", desifre_dizi[sayac]);                                      //Desifrelenmii¿% metin ekrana yazdiriliyor.
        sayac++;
    }
}
```

Figüre 3.13 Deşifreleme Fonksiyonu

Deşifreleme fonksiyonu şifreleme fonksiyonun ters mantığıyla çalışmaktadır.

Öncelikle diziden okunan karakterler matriste bulunup kullanıcıdan alınmış satır sütun mesafe bilgisine göre deşifreleme işlemi gerçekleştirilmiştir.

Deşifre edilen karakterler sırayla ekrana bastırılarak programımız sonlandırılmıştır.

SONUÇ

İki görevi de iyice analiz ettikten sonra vardığımız sonuç şu yöndedir :
Şifreleme algoritmaları günümüzde iletişim başta olmak üzeri bir çok alanda kullanılmaktadırlar. Bu algoritmaların önemi kuşkusuz ortadır. Bir şifremele algoritması ortaya koyarken iki önemli şey mevcuttur. Birincisi kırılmasının zorlaşıtırılması, ikincisi ise sisteme veya aktarım ortamına getireceği yük. Bu ikisi genellikle birbirinin zıttı çalışmaktadır. Ne kadar zor bir sistem kurarsak o kadar yük artmaktadır. Bu iki parametreyi de göze alınarak bu algoritmalar geliştirilebilir.

KAYNAKLAR

[1] SSD.EFF.org, “Şifreleme Hakkında Bilinmesi Gerekenler” ,2018

[2] H İrem Türkmen ÇİLİNGİR, Yıldız Teknik Üniversitesi, Bilgisayar Mühendisliği, “Programlama Dilleri Ders Notları” ,2020

[3] H İrem Türkmen ÇİLİNGİR, Yıldız Teknik Üniversitesi, Bilgisayar Mühendisliği, “Programlama Dilleri Proje Notları” ,2020

[4] Bilim Çocuk Dergisi, “Acaba Ne Yazıyor?”

Grup İçi İş Bölümü

C dosyalarının oluşturulması :

MatrisOlustur.c → Uğur Kelleci ,Emre Kaan Özkan

MetinSifrele.c → Emre Kaan Özkan

MetinSifreCozme.c→Uğur Mestçi, Uğur Kelleci

Görev 2 sezarMatris.c →Emre Kaan Özkan Uğur Mestçi Uğur Kelleci,

2. algoritmanını kurulması → Uğur Mestçi, Emre Kaan Özkan

Rapor Hazırlanması :

Rapor hazırlanırken herkes kendi kodunu raporda açıklamıştır.

Şifre Matrisi kısmını Uğur Kelleci, Şifreleme kısmını Emre Kaan Özkan, Deşifreleme kısmı ise Uğur Mestçi tarafından hazırlanmıştır.

Rapor Düzenlenmesi → Emre Kaan Özkan

C programları oluşturulurken hep beraber Discord ortamında ortak çalışmalar yürütülmüştür.