YILDIZ TEKNİK ÜNİVERSİTESİ

ELEKTRİK-ELEKTRONİK FAKÜLTESİ

BİLGİSAYAR MÜHENDİSLİĞİ

BLM3021 – ALGORİTMA ANALİZİ

1.ÖDEV 2.SORU RAPORU

Konu : Von Neumann’s Neighborhood Kuralını Gerçekleme

Ad Soyad : Eren TUTUŞ

Okul No : 17011702

E-posta : l1117702@std.yildiz.edu.tr

Programın Çalışma Süreci

1-) int *vonNeumann (int **matris , int matrisBoyutu , int n) Fonksiyonu :

Von Neumann Neighborhood kuralı ({(x,y) : |x-x0| + |y-y0| <= n}) doğrultusunda matristeki siyah karelerin satır numarası ile matrisin ortadaki elemanın satır numarası, arasındaki farkın mutlak değeri ile sütun numaraları arasındaki farkın mutlak değerinin toplamı N değerinden küçük veya eşit olmalı.

Bu koşulu gerçekleştirmek üzere öncelikle sayici dizisi oluşturduk bu dizi için hafızada yer açtık ve içindeki değerleri 0 yaptık bunun sebebi RAM'den rastgele gelen değerler diziye atacağımız değerler ile karışmasın diye. Daha sonra yukarıdaki tanımladığımız neighborhood kuralını if sorgusu ile gerçekleyip matris[i][j] = 1 işlemi yaparak yani siyah kareleri 1 olarak işaretleyerek aynı zamanda sayici dizimizin indislerine her satırdaki siyah kare sayısınıda aktarmış oluyoruz.

Son olarakta işlemler tamamlanınca her satırdaki siyah kare sayısının yer aldığı dizi olan sayici dizisini döndürüyoruz.

2-) main( ) Fonksiyonu :

Main fonksiyonunda öncelikle kullanıcıdan bir n değeri alıyoruz ve bu n değerine göre matris boyutunu 2*n+3 formülü ile AxA boyutunda bir matris oluşturacağımızı belirtmiş oluyoruz. Çok boyutlu matris oluşturuyoruz ve her satır için bellekte yer açıyoruz. Aynı şekilde her satır için gerekli olan sütun sayısı için bellekte yer açıyoruz. Daha sonra, örneğin AxA lık bir matris oluşturulduğunda içinde tuhaf değerler yazmaması için her indisi 0 olarak işaretliyoruz.

VonNeumann fonksiyonuna gönderdiğimiz parametreler sonucu gerekli işlemler yapıldıktan sonra döndürülen sayici dizisini, main fonksiyonunda oluşturduğumuz diğer bir sayici dizisine aktarıyoruz. Daha sonra yeni oluşacak olan matrisimizi ekrana yazdırıyoruz. Ayrıca, her satırdaki siyah kare sayısını alt alta olacak şekilde yazdırıp, en sonunda ise toplamda kaç tane siyah kare oluşmuşsa bunuda ekrana yazdırıyoruz.

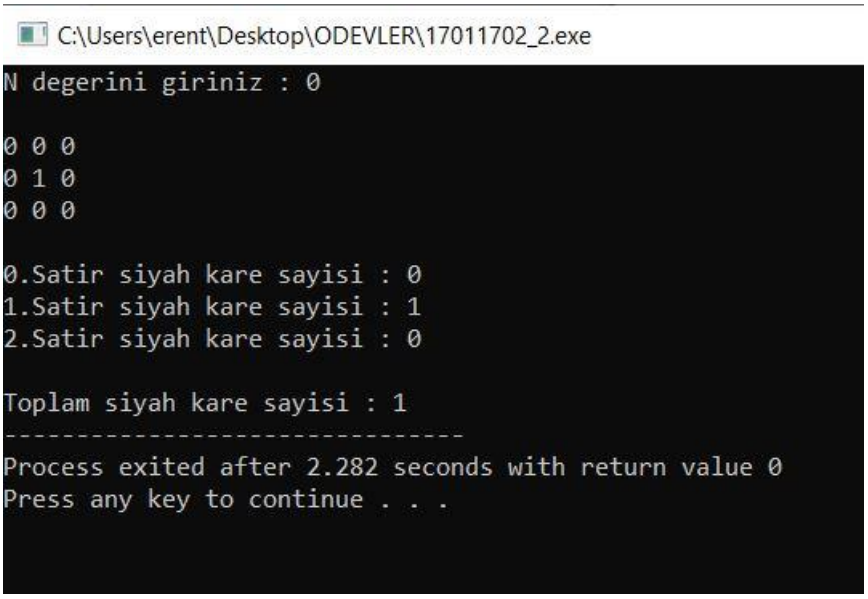
Son olarak ise matrisi hafızadan silmemiz gerektiği için satırları boşaltıyoruz. Satırları boşalttıktan sonra free (matris) ile matrisi tamamen ortadan kaldırıyoruz.

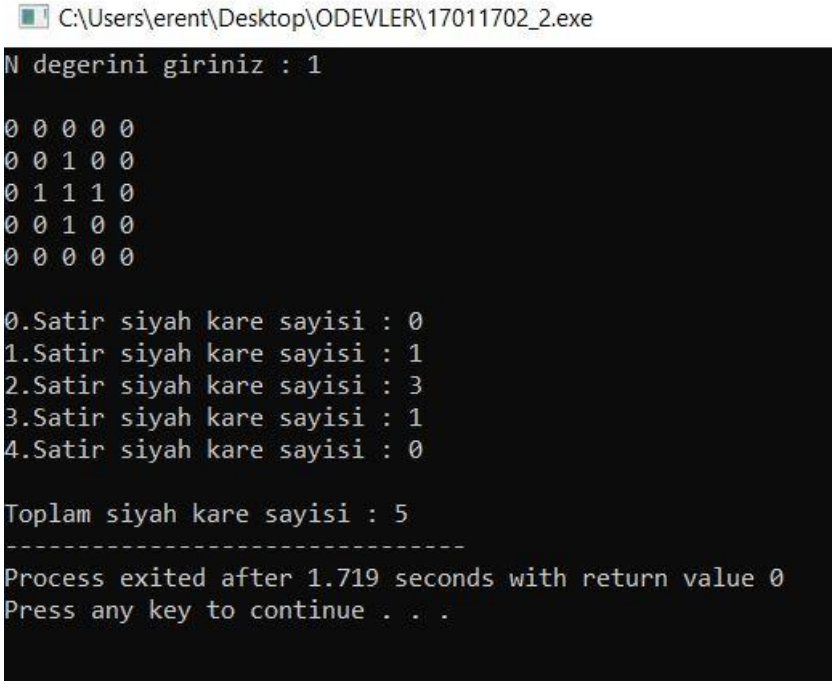
C KODU

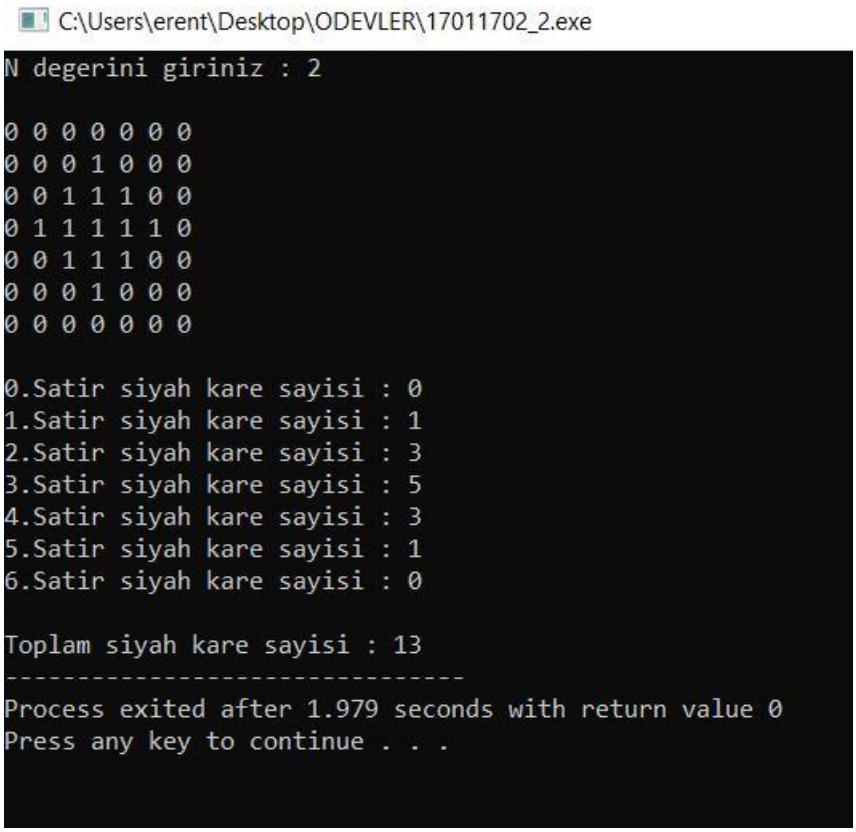
```c
#include <stdio.h>
#include <stdlib.h>


int *VonNeumann(int **matris, int matrisBoyutu,int n){   // main'den gelen matris,matris boyutu ve kullanıcıdan alınan n değeri parametre olarak gelmiştir.
    int *sayici; // Her satırda siyah renk olan karelerin sayısını dizimizin indislerine atabilmek için "sayici" isimli bir dizi oluşturuyoruz.
    int i; // Döngü için sayac degiskeni
    int j; // Döngü için sayaç değişkeni
    sayici = (int *)malloc(sizeof(int)*matrisBoyutu);   // her satırın siyah kare sayısını atabilmek için bellekte yer açıyoruz.
    for (i=0;i<matrisBoyutu;i++){
            sayici[i] = 0;      // Matris boyutu kadar içerisini 0 değeriyle dolduruyoruz bunun nedeni tuhaf değerler gelmesini önlemek.
    }
    for (i=0;i<matrisBoyutu;i++){   // {(x,y) : |x-x0| + |y-y0| <= n} koşulunu gerçekledik.
        for (j=0;j<matrisBoyutu;j++){ // Bu kural doğrultusunda; matristeki siyah karelerin satır numarası ile matrisin ortadaki elemanın satır numarası,
            if (abs(i-n-1)+abs(j-n-1)<=n){  // arasındaki farkın mutlak değeri ile sütun numaraları arasındaki farkın mutlak değerinin toplamı N değerinden küçük veya eşit olmalı.
                matris[i][j] = 1;   // Matrisin i. satırını ve j.sütununu 1 olarak işaretledik.
                sayici[i]++;   // ve ilgili satırda kaç tane 1(siyah kare) varsa bunların toplamını aldık.
            }
        }
    }
    return sayici;   // Her satırdaki siyah kare sayısının yer aldığı diziyi döndürüyoruz.
}

int main() {

    int n;  // Kullanıcıdan alınacak değer
    printf("N degerini giriniz : ");
    scanf("%d", &n);  // Kullanıcıdan alınan değer n değişkenine atanıyor.
    int matrisBoyutu;  AxA ' lık bir matris için n değeri ile şekildeki örüntü arasında bağ kuruyoruz.
    matrisBoyutu = 2*n+3;  // n = 0 için; 3x3 matris , n = 1 için; 5x5 matris , n = 2 için; 7x7 matris ...
    int i;  // döngü için kullanılacak sayaç değişkeni
    int j;  // döngü için kullanılacak sayaç değişkeni
    int **matris;  //çok boyutlu matrisimizi oluşturuyoruz.
    int *sayici; // Her satırda siyah renk olan karelerin sayısını dizimizin indislerine atabilmek için "sayici" isimli bir dizi oluşturuyoruz.

    matris = (int **) malloc(sizeof(int*)*matrisBoyutu); // Her satır bellekte yer açıyoruz.
    if(matris == NULL){
        printf("Yetersiz bellek !");
```

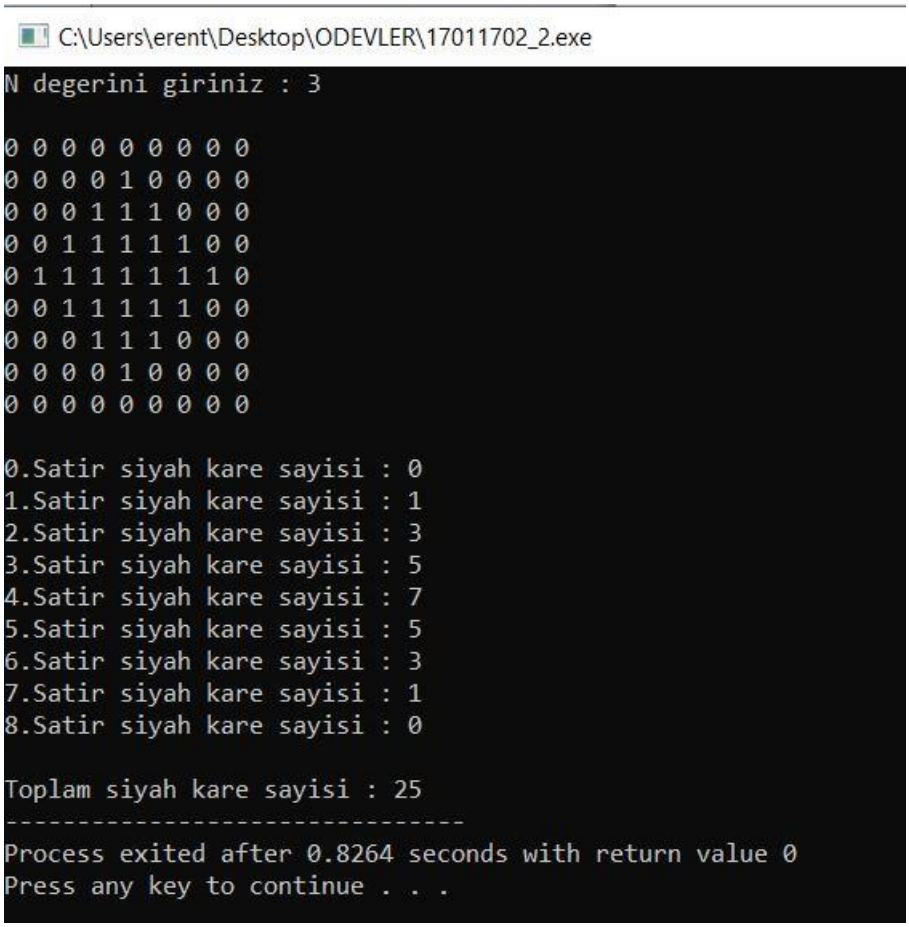
```
    }

    // Her satır için gerekli olan sütun sayısı için bellekte yer açıyoruz.
    for(i=0;i<matrisBoyutu;i++){
        matris[i] = (int *)malloc(sizeof(int)*matrisBoyutu );
        if(matris[i] == NULL){
            printf("Yetersiz bellek !");
        }
    }

    // Matris içindeki tüm değerleri 0 yapıyoruz. Çünkü biz matrisde
gerekli alanları 1 yapacağız ve kalanlar 0 olacak.
    for (i=0;i<matrisBoyutu;i++){
        for (j=0;j<matrisBoyutu;j++){
            matris[i][j] = 0;
        }
    }

    printf("\n");

    sayici = VonNeumann(matris,matrisBoyutu,n);   // // Her satırdaki siyah
kare sayısının yer aldığı diziyi döndürdük ve sayac dizimize attık.

    for (i=0;i<matrisBoyutu;i++){
        for (j=0;j<matrisBoyutu;j++){
            printf("%d ",matris[i][j]);   // AxA ' lık matrisimizde 1(siyah
kare) olan kısımları göstermek için matrisi ekran yazdırdık.
        }
        printf("\n");
    }

    printf("\n");

    for (i=0;i<matrisBoyutu;i++){   // Her satırda kaç tane 1(siyah kare)
varsa bunların sayısı ekrana yazdırıyoruz.
        printf("%d.Satir siyah kare sayisi : %d\n",i,sayici[i]);
    }

    // Ve matrisdeki toplam siyah karesini ekrana yazdırıyoruz.
    printf("\nToplam siyah kare sayisi : %d",(2*n*(n+1))+1);

    // Burada matrisi hafızadan silmemiz gerektiği için satırları
boşaltıyoruz.
    for(i=0;i<matrisBoyutu;i++) {
        free(matris[i]);
    }

    // Satırları boşalan matrisi boş olarak ifade ediyoruz.
    free(matris);

    return 0;
}
```

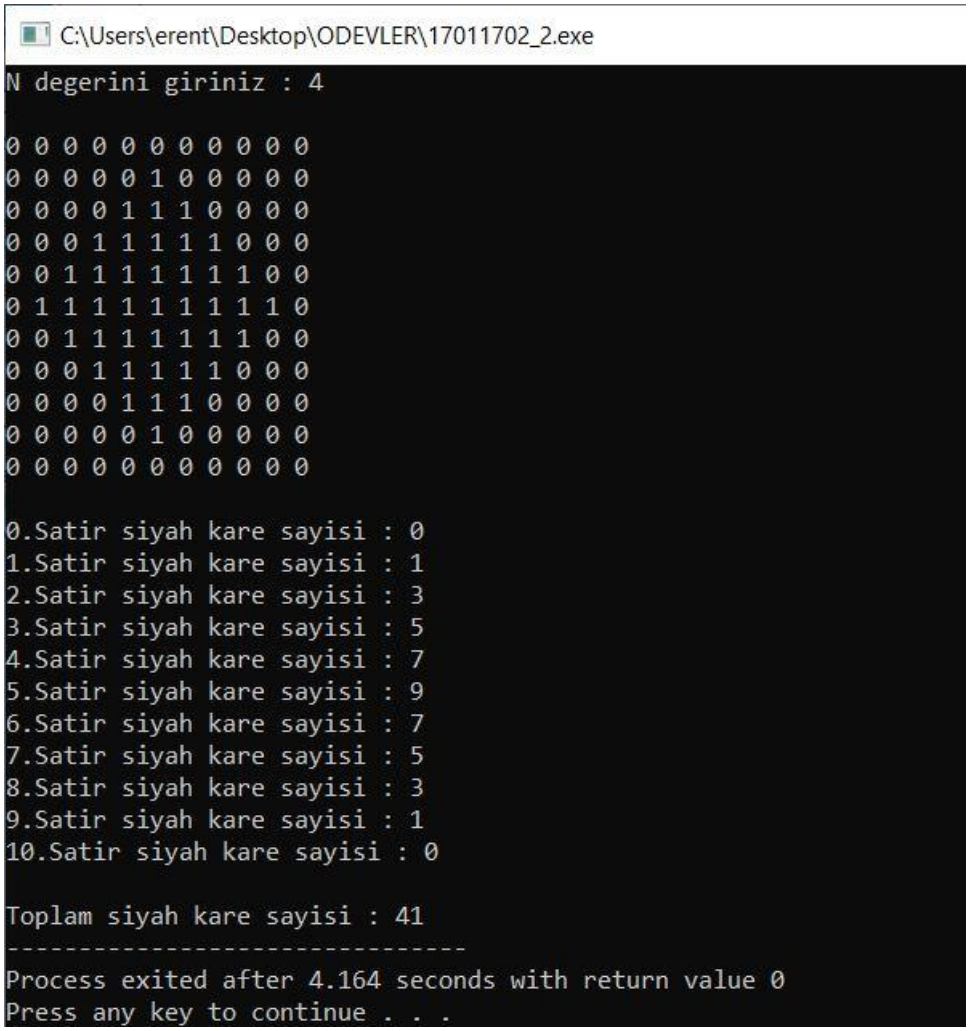
EKRAN ÇIKTILARI






C:\Users\erent\Desktop\ODEVLER\17011702_2.exe
N degerini giriniz : 2
0 0 0 0 0 0 0
0 0 0 1 0 0 0
0 0 1 1 1 0 0
0 1 1 1 1 1 0
0 0 1 1 1 0 0
0 0 0 1 0 0 0
0 0 0 0 0 0 0
0.Satir siyah kare sayisi : 0
1.Satir siyah kare sayisi : 1
2.Satir siyah kare sayisi : 3
3.Satir siyah kare sayisi : 5
4.Satir siyah kare sayisi : 3
5.Satir siyah kare sayisi : 1
6.Satir siyah kare sayisi : 0
Toplam siyah kare sayisi : 13
Process exited after 1.979 seconds with return value 0
Press any key to continue . . .


C:\Users\erent\Desktop\ODEVLER\17011702_2.exe
N degerini giriniz : 3
0 0 0 0 0 0 0 0 0
0 0 0 0 1 0 0 0 0
0 0 0 1 1 1 0 0 0
0 0 1 1 1 1 1 0 0
0 1 1 1 1 1 1 1 0
0 0 1 1 1 1 1 0 0
0 0 0 1 1 1 0 0 0
0 0 0 0 1 0 0 0 0
0 0 0 0 0 0 0 0 0
0.Satir siyah kare sayisi : 0
1.Satir siyah kare sayisi : 1
2.Satir siyah kare sayisi : 3
3.Satir siyah kare sayisi : 5
4.Satir siyah kare sayisi : 7
5.Satir siyah kare sayisi : 5
6.Satir siyah kare sayisi : 3
7.Satir siyah kare sayisi : 1
8.Satir siyah kare sayisi : 0
Toplam siyah kare sayisi : 25
Process exited after 0.8264 seconds with return value 0
Press any key to continue . . .


C:\Users\erent\Desktop\ODEVLER\17011702_2.exe
N degerini giriniz : 4
0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 1 0 0 0 0 0
0 0 0 0 1 1 1 0 0 0 0
0 0 0 1 1 1 1 1 0 0 0
0 0 1 1 1 1 1 1 1 0 0
0 1 1 1 1 1 1 1 1 1 0
0 0 1 1 1 1 1 1 1 0 0
0 0 0 1 1 1 1 1 0 0 0
0 0 0 0 1 1 1 0 0 0 0
0 0 0 0 0 1 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0
0.Satir siyah kare sayisi : 0
1.Satir siyah kare sayisi : 1
2.Satir siyah kare sayisi : 3
3.Satir siyah kare sayisi : 5
4.Satir siyah kare sayisi : 7
5.Satir siyah kare sayisi : 9
6.Satir siyah kare sayisi : 7
7.Satir siyah kare sayisi : 5
8.Satir siyah kare sayisi : 3
9.Satir siyah kare sayisi : 1
10.Satir siyah kare sayisi : 0
Toplam siyah kare sayisi : 41
Process exited after 4.164 seconds with return value 0
Press any key to continue . . .