Yıldız Teknik Üniversitesi
Elektrik-Elektronik Fakültesi
Bilgisayar Mühendisliği Bölümü

BLM1012
Yapısal Programlamaya Giriş
Gr:2
Öğr.Gör.Dr. Ahmet ELBİR
Dönem Projesi

İsim: Konuralp BOL

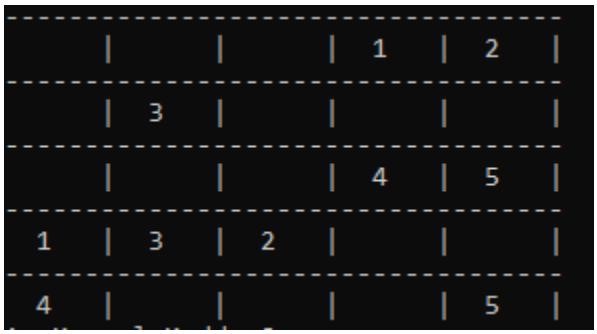
No:22011618

E-posta:konuralp.bol@std.yildiz.edu.tr

İçindekiler

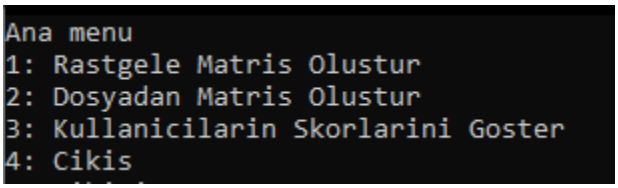
Ön bilgi

Program bir matris üzerinde sayı eşleştirme oyunudur. Matriste sayılar çiftler halinde bulunur ve oyuncu bu sayıları 4 yönde ilerleyerek eşleştirecek yolları çizer. Bu programda oyuncu sadece manuel olarak oynayabilir otomatik oynama özelliği yoktur.



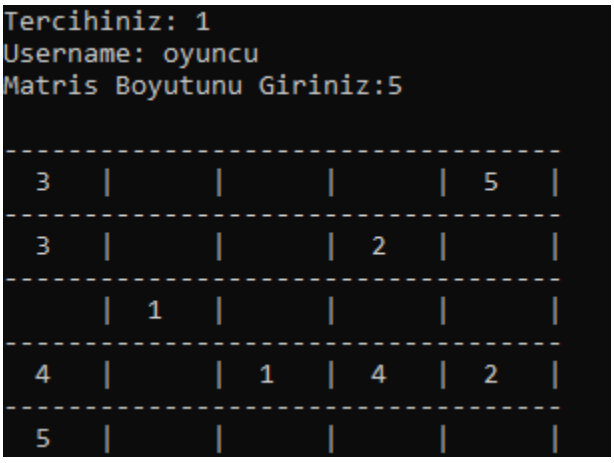
Ana Menü

Kullanıcı ana menüde matrisin rastgele oluşturulacağını veya bir dosyadan okunacağını söyler.

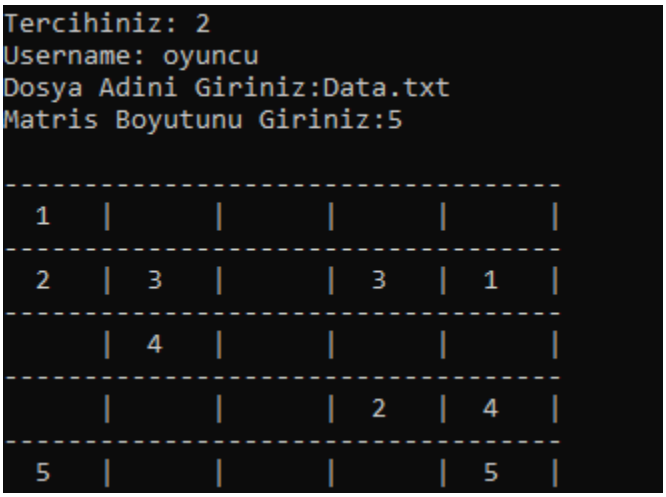


Matris oluşturma seçeneği seçildikten sonra kullanıcıdan kullanıcı adı istenir ve oluşturulacak matrisin boyutu istenir. Eğer matris dosyadan okunacaksa dosyanın adı da istenir.

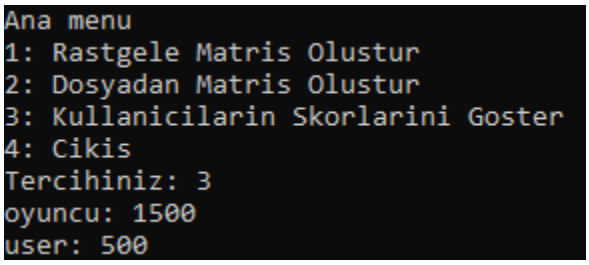
Rastgele Matris



Dosyadan Okunmuş Matris

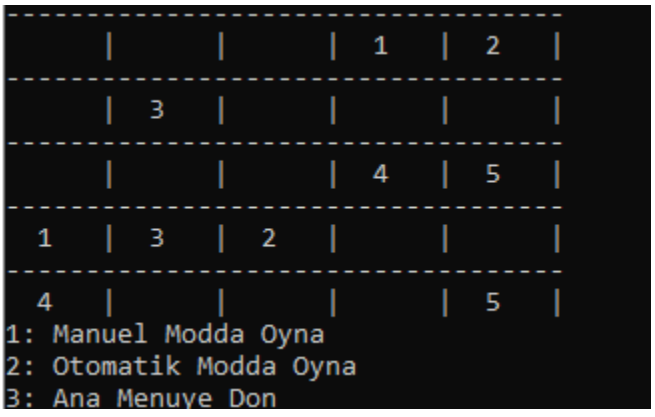


Kullanıcıların Skorları

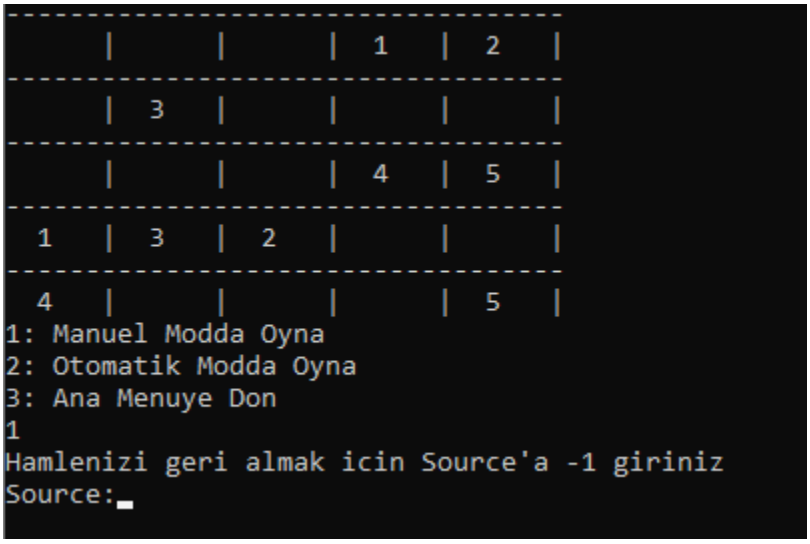


Oyunu Oynama Çeşitleri

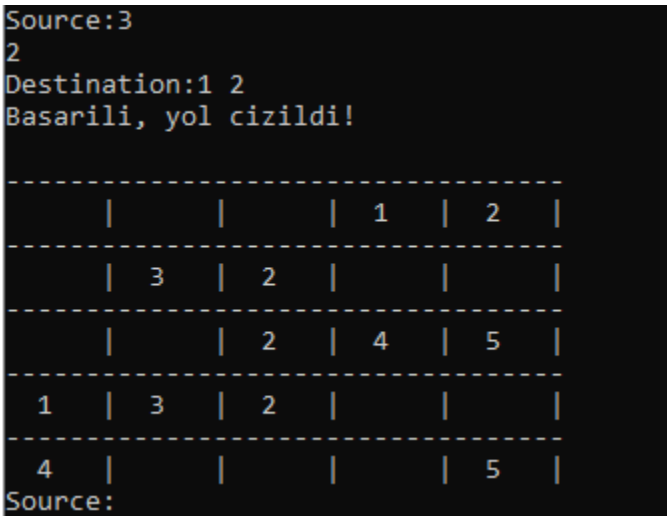
Menüde yazmasına ragmen kullanıcı oyunu sadece manuel olarak oynayabilir. Otomatik oynama özelliği yoktur.

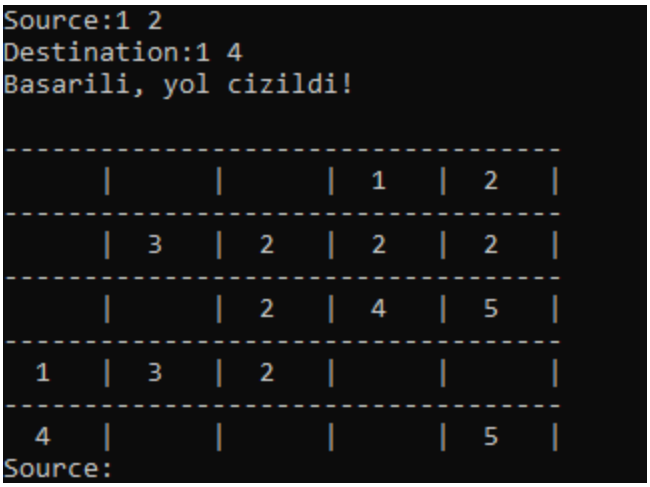


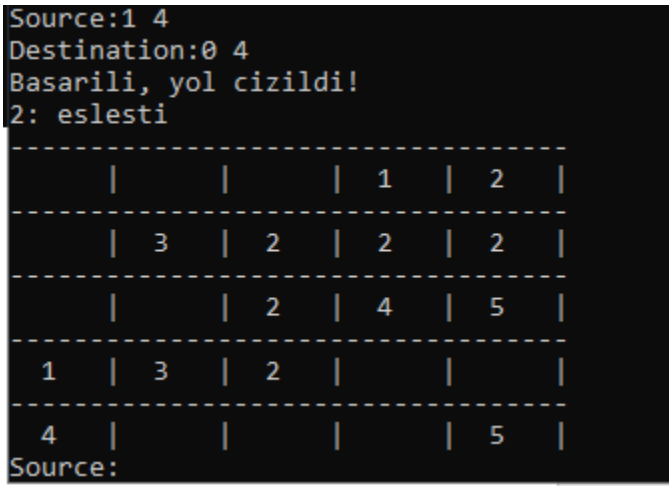
Manuel Mod



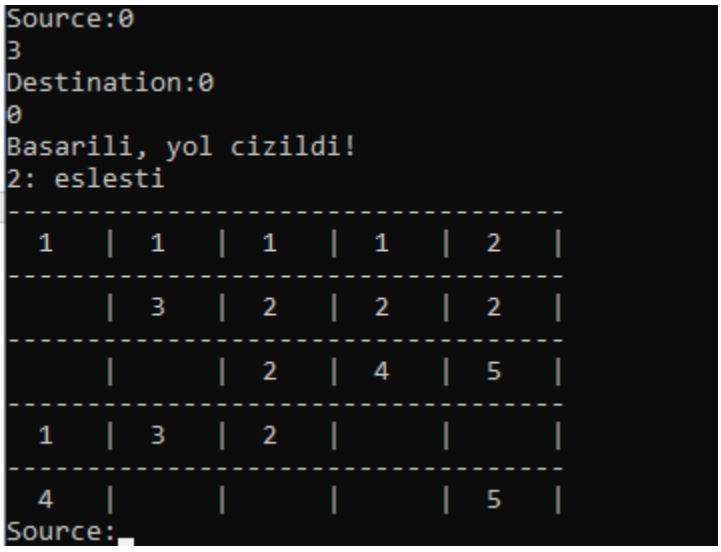
Kullanıcıdan başlangıç noktası ve gidiş noktası istenir. Eğer kullanıcı yaptığı hamleyi geri almak istiosa -1 girişi yapar.







2'nin eşleştiği yazmakta.



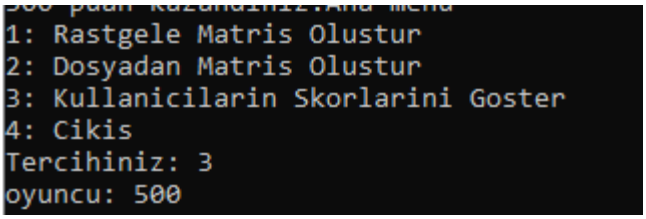
```
Source:0 0
Destination:3 0
Basarili, yol cizildi!
1: eslesti 2: eslesti
-----------------------------------
  1   |  1   |  1   |  1   |  2   |
-----------------------------------
  1   |  3   |  2   |  2   |  2   |
-----------------------------------
  1   |      |  2   |  4   |  5   |
-----------------------------------
  1   |  3   |  2   |      |      |
-----------------------------------
  4   |      |      |      |  5   |
Source:
```

```
Source:1 1
Destination:3 1
Basarili, yol cizildi!
1: eslesti 2: eslesti 3: eslesti
-----------------------------------
  1   |  1   |  1   |  1   |  2   |
-----------------------------------
  1   |  3   |  2   |  2   |  2   |
-----------------------------------
  1   |  3   |  2   |  4   |  5   |
-----------------------------------
  1   |  3   |  2   |      |      |
-----------------------------------
  4   |      |      |      |  5   |
Source:
```

```
Source:4 0
Destination:4 3
Basarili, yol cizildi!
1: eslesti 2: eslesti 3: eslesti
-----------------------------------
  1   |  1   |  1   |  1   |  2   |
-----------------------------------
  1   |  3   |  2   |  2   |  2   |
-----------------------------------
  1   |  3   |  2   |  4   |  5   |
-----------------------------------
  1   |  3   |  2   |      |      |
-----------------------------------
  4   |  4   |  4   |  4   |  5   |
Source:
```

```
Source:4 3
Destination:2 3
Basarili, yol cizildi!
1: eslesti 2: eslesti 3: eslesti 4: eslesti
-----------------------------------
  1   |  1   |  1   |  1   |  2   |
-----------------------------------
  1   |  3   |  2   |  2   |  2   |
-----------------------------------
  1   |  3   |  2   |  4   |  5   |
-----------------------------------
  1   |  3   |  2   |  4   |      |
-----------------------------------
  4   |  4   |  4   |  4   |  5   |
Source:
```

```
Source:4 4
Destination:2 4
Basarili, yol cizildi!
1: eslesti 2: eslesti 3: eslesti 4: eslesti 5: eslesti
-----------------------------------
  1   |  1   |  1   |  1   |  2   |
-----------------------------------
  1   |  3   |  2   |  2   |  2   |
-----------------------------------
  1   |  3   |  2   |  4   |  5   |
-----------------------------------
  1   |  3   |  2   |  4   |  5   |
-----------------------------------
  4   |  4   |  4   |  4   |  5   |
500 puan kazandiniz.Ana menu
```



Tüm sayılar eşleştiğinde oyun biter ve kullanıcının kazandığı puan yazdırılır. Kullanıcı ana menüye dönüp tüm kullanıcıların puanına bakabilir.

Zaman Karmaşıklığı

Manuel mod için zaman karmaşıklığını hesaplayacak olursak.

Her hamle başına:

drawRoad fonksiyonu çağrılır. karmaşıklığı: O(n)'dir.

Eğer kullanıcı yol çizmek yerine undo yapmak isterse bunun da karmaşıklığı O(n)'dir.

gameHistory bir sağa kaydırılır karmaşıklığı sabit olduğundan: O(1)'dir.

Pairler birbiri ile eşleşmiş mi diye kontrol edilir, karmaşıklığı:4*N yani O(N)'dir.

drawboard fonksiyonu çağırılır, karmaşıklığı: $O(n^2)$dir.

Zaman karmaşıklığı için en yüksek önceliğe sahip değer drawboard fonksiyonuna aittir. Bu yüzden Manuel modun karmaşıklığına: $O(hamleSayisi*n^2)$ diyebiliriz.

Kod Analizi

Öncelikle programın başında kullanıcıdan matrisin nasıl oluşturulacağını alıp, eğer dosyadansa dosyadan matrisi alıyorum. Bu işlemden önce yazdığım zeroFill adındaki fonksiyonla matrisi temizliyorum ki önceki oyunlar diğer oyunları etkilemesin.

```
do{
    printf("Ana menu\n1: Rastgele Matris Olustur\n2: Dosyadan Matris Olustur\n3: Kullanicilarin Skorlarini Goster\n4: Cikis\nTercihiniz: ");
    random = 0;
    scanf("%d", &choice);
    if(choice == 1 || choice == 2){
        playerIndex = playerInit(players,&users);
    }

    if(choice == 1){
        printf("Matris Boyutunu Giriniz:");
        scanf("%d", &N);
        zeroFill(matris,N);
        random = 1;
        for(i=0;i<N;i++){
            for(j=0;j<2;j++){
                do{
                    x = rand() % N;
                    y = rand() % N;
                }while(matris[x][y] != 0);/*ciftleri pairs matrisine atiyor*/
                matris[x][y] = i+1;
                pairs[j][0][i] = x;
                pairs[j][1][i] = y;
            }
        }
        drawBoard(matris,N);
    }
```

```
    else if(choice == 2){
        printf("Dosya Adini Giriniz:");
        scanf("%s",fileName);
        printf("Matris Boyutunu Giriniz:");
        scanf("%d", &N);
        zeroFill(matris,N);
        readFromFile(matris, fileName);
        drawBoard(matris,N);
        for(i=0;i<N;i++){
            for(j=0;j<2;j++){
                for(k=0;k<2;k++){
                    pairs[j][k][i] = -1;
                }
            }
        }

        for(i=0;i<N;i++){
            for(j=0;j<N;j++){
                val = matris[i][j] - 1;
                if(val != -1){
                    if(pairs[0][0][val] == -1 && pairs[0][1][val] == -1){
                        pairs[0][0][val] = i;
                        pairs[0][1][val] = j;
                    }
                    else{
                        pairs[1][0][val] = i;
                        pairs[1][1][val] = j;
                    }
                }
            }
        }
    }
```

Aynı şekilde pairsi de -1 ile dolduruyorum ki karışmasın. Kullanıcıdan matrisi aldıktan veya rastgele oluşturduktan sonra pairslerin nerede olduğunu matrisi tarıyarak buluyorum ve pairs'e atıyorum.

Pairslerin yerini kullanıcının oyunda genel olarak kuralların dışına çıkmasını engellemek ve oyunun bitip bitmediğinin kontrolunu yapmak için tutuyorum.

```
    if(choice == 1 || choice == 2){
        printf("\n1: Manuel Modda Oyna\n2: Otomatik Modda Oyna\n3: Ana Menuye Don\n");
        scanf("%d",&choice);
        if(choice == 1){
            manualPlay(matris,N,gameHistory, pairs,scores,random,users,playerIndex,players);
        }
        if(choice == 2){
            printf("Otomatik oynama ozelligi mevcut degildir.\n");
        }
    }
```

Kullanıcı oyun modunu seçiyor. Sadece manuel modu bulunmakta. Bu yüzden otomatik seçildiğinde uyarı veriyor.

```
void manualPlay(int matris[MAX][MAX],int N, int gameHistory[2][2][6], int pairs[2][2][MAX],int sc
    int history=0,fromX,fromY,f,toX,toY,val,undoCount=0,pairSelect,i;
    printf("Hamlenizi geri almak icin Source'a -1 giriniz");
    do{


    printf("\nSource:");
    scanf("%d",&fromX);
    if(fromx == -1){
        if(history >0){
            f=1;
            history--;
            undo(matris,pairs,gameHistory,1,history);
            drawBoard(matris,N);
            undoCount++;
        }
    }
    else{
    f = 0;
    scanf("%d",&fromY);
    printf("Destination:");
    scanf("%d",&toX);
    scanf("%d",&toY);
    val = matris[fromX][fromY] - 1;
```

Burada kullanıcı eğer undo yapmak isterse -1 girmesi gerekiyor böylece undo fonksiyonu çalışıyor.

Kullanıcıdan nereden nereye gitmesine dair parametreler isteniyor.

```
if(((pairs[0][0][val] == fromX &&  pairs[0][1][val] == fromY) || (pairs[1][0][val] == fromX &&  pairs[1][1][val] == fromY)) && (toX < N && toY < N && toX>=0 && toY>=0)){
    if(pairs[0][0][val] == fromX &&  pairs[0][1][val] == fromY){/*Hangi pairin hareket edecegini buluyor.*/
        pairSelect = 0;
    }
    else if(pairs[1][0][val] == fromX &&  pairs[1][1][val] == fromY){
        pairSelect = 1;

    }
    if(drawRoad(matris,fromX,fromY,toX,toY)){
        printf("Basarili, yol cizildi!\n");

        pairs[pairSelect][0][val] = toX;
        pairs[pairSelect][1][val] = toY;
        for(i=history;i>0;i--){/*Historyi saga kaydiriyor*/
            gameHistory[0][0][i] = gameHistory[0][0][i-1];
            gameHistory[0][1][i] = gameHistory[0][1][i-1];
            gameHistory[1][0][i] = gameHistory[1][0][i-1];
            gameHistory[1][1][i] = gameHistory[1][1][i-1];
        }/*Historye kaydediyor*/
        gameHistory[0][0][0] = fromX;
        gameHistory[0][1][0] = fromY;
        gameHistory[1][0][0] = toX;
        gameHistory[1][1][0] = toY;
        if(history!=5){
            history++;
        }
    }
    else{
        printf("Hatali hamle");
    }
```

Başta verilen parametrelerin uygunluğu(matrisin dışına taşma,çapraz gitme gibi durumlar) kontrol ediliyor. Eğer uygunsa pairSelect ile çiftlerden hangisinin hareket edeceği bulunuyor. Bunu yapma sebebim hareket eden çifti yeni konumu ile güncellemek. Daha sonra drawRoad fonksiyonuna verilen parametreler giriliyor.Eğer yol çizildiyse historye hareketleri ekliyor.

```
int drawRoad(int matris[][MAX], int fromX, int fromY, int toX, int toY){ /*Oyuncunun girdigi degerlere gore yol cizer*/
    int i,val = matris[fromX][fromY];
    if((matris[toX][toY] == 0 || val == matris[toX][toY]) && (fromX == toX || fromY == toY)){/*Kulanicinin varmaya calistigi
        if(fromX == toX){/*hangi duzlemde hareket edilecegini buluyor.*/
            if(fromY<toY){
                for(i=fromY+1;i<toY;i++){
                    if(matris[fromX][i]!=0){/*Arada 0dan farkli deger varsa cizmeyip 0 donduruyor */
                        return 0;
                    }
                }
                for(i=fromY+1;i<toY+1;i++){
                        matris[fromX][i] = val;
                }
            }
            else{
                for(i=toY+1;i<fromY;i++){
                    if(matris[fromX][i]!=0){
                        return 0;
                    }
                }
                for(i=toY;i<fromY+1;i++){
                        matris[fromX][i] = val;
                }
            }
        }
```

```
        else if(fromY == toY){
            if(fromX<toX){
                for(i=fromX+1;i<toX;i++){
                    if(matris[i][fromY]!=0){
                        return 0;
                    }
                }
                for(i=fromX+1;i<toX+1;i++){
                        matris[i][fromY] = val;
                }
            }
            else{
                for(i=toX+1;i<fromX;i++){
                    if(matris[i][fromY]!=0){
                        return 0;
                    }
                }
                for(i=toX;i<fromX+1;i++){
                        matris[i][fromY] = val;
                }
            }
        }
        else{
            return 0;
        }
        return 1;
    }
    else{
        return 0;
    }
}
```

drawRoad fonksiyonu bir int fonksiyonudur ve 0 döndürürse verilen parametreler ile bir yol çizilemiyor demektir. 1 döndürdüyse yol başarı ile çizilmiştir. drawRoad önce yolun yönünü belirler daha sonra gidilecek yöndeki başlangıç veya varış değerlerinden hangisi daha yüksek onu belirler ve buna göre yolun boş olup olmadığını kontrol ettikten sonra yolu çizer.

```
void undo(int matris[][MAX], int pairs[2][2][MAX], int h[][2][6], int count,int history){
    int i,val,pairSelect,fromX,fromY,toX,toY;
    val = matris[h[0][0][0]][h[0][1][0]]-1;

    fromX = h[0][0][0];
    fromY = h[0][1][0];
    toX = h[1][0][0];
    toY = h[1][1][0];
    if(pairs[0][0][val] == toX &&  pairs[0][1][val] == toY){
        pairSelect = 0;
    }
    else if(pairs[1][0][val] == toX &&  pairs[1][1][val] == toY){
        pairSelect = 1;

    }

    if(fromX == toX){
        if(fromY<toY){
            for(i=fromY+1;i<toY+1;i++){
                matris[fromX][i] = 0;
            }
        }
        else{
            for(i=toY;i<fromY;i++){
                matris[fromX][i] = 0;

            }
        }
    }
    else if(fromY == toY){
        if(fromX<toX){
            for(i=fromX+1;i<toX+1;i++){

                matris[i][fromY] = 0;

            }
        }
        else{
            for(i=toX;i<fromX;i++){
                matris[i][fromY] = 0;
            }
        }
    }
```

```
    if(pairs[0][0][val] == pairs[1][0][val] && pairs[0][1][val] == pairs[1][1][val]){
        matris[pairs[0][0][val]][pairs[0][1][val]] = val+1;
        matris[pairs[1][0][val]][pairs[1][1][val]] = val+1;
    }
    pairs[pairSelect][0][val] = fromX;
    pairs[pairSelect][1][val] = fromY;
    for(i=0;i<history;i++){
        h[0][0][i] = h[0][0][i+1];
        h[0][1][i] = h[0][1][i+1];
        h[1][0][i] = h[1][0][i+1];
        h[1][1][i] = h[1][1][i+1];
    }
}
```

Undo fonksiyonu historyden aldığı değerler ile drawRoadın yaptığını 0 için yapıyor ve historyden o değerleri siliyor.

```
        for(i=0;i<N;i++){/*pairler birbiriyle eslesiyor mu kontrol ediliyor.*/
            if(pairs[0][0][i] != pairs[1][0][i] || pairs[0][1][i] != pairs[1][1][i]){
                f=1;
            }
            else{
                printf("%d: eslesti ", i+1);
            }
        }

    }
    else{
        printf("Hatali hamle");
        f=1;
    }
    drawBoard(matris,N);
    }
    }while(f);
    scoreCalculator(players,users,scores,playerIndex,random,undoCount,N);
}
```

manualPlayin kalan kısmında her hareketten sonra pairler birbiriyle eşleşmiş mi kontrol ediliyor. Eşleşenler ekrana yazdırılıyor. Eğer hepsi eşleşti ise oyun bitiyor ve scoreCalculatora parametreler giriliyor.

```
void scoreCalculator(char players[MAX][MAX], int users, int scores[MAX], int playerIndex,int random, int undoCount,int N)
    int score = 100;

    score *= N;

    if(random == 1){
        score *= 2;
    }
    score -= undoCount * N;
    scores[playerIndex] += score;
    printf("\n%d puan kazandiniz.", score);

}
```

Bu fonksiyon görüldüğü gibi şu mantıkla skor hesaplıyor:

Base skorumuz: 100

Skor matrisin büyüklüğü ile çarpılıyor

Eğer matris rastgele üretildi ise skor 2 ile çarpılıyor

Undo yapıldıysa skor undo sayısının n ile çarpılması kadar azaltılıyor.

Bu skoru scores arrayine kaydediyor.

```c
int playerInit(char players[MAX][MAX],int* users){/*Oyuncu ismi daha once
    char playerName[MAX];
    int n,i,r,j;
    printf("Username: ");
    scanf("%s", playerName);
    n=0;
    for(i=0;i<*users;i++){
        r=1;
        j=0;

        while(players[j][i] != '\0'){
            if(players[j][i] != playerName[j]){
                r=0;
            }
        j++;
        }
        if(r==1 && playerName[j] == '\0' && players[j][i] == '\0'){
            return i;
        }
    }

    if(n==0){
        strcpy(players[*users], playerName);
        (*users)++;
        return (*users)-1;

    }
    return 0;
}
```

playerInit ise kullanıcı isminin daha önce var olup olmadığına bakıp. Eğer daha önce girilen adda bir kullanıcı oynadı ise o kullanıcının indexini veriyor. Eğer yeni bir kullanıcı ise yeni bir indexe o kullanıcıyı atıyor.

```c
void userScores(char players[MAX][MAX], int users, int scores[MAX]){/*Oyuncu puanlarini bastiran fonksiyon*/
    int i;
    for(i=0;i<users;i++){
        printf("%s: %d\n", players[i], scores[i]);
    }


}
```

```c
void zeroFill(int matris[MAX][MAX], int N){/*Bir matrisi 0'la dolduran fonksiyon*/
    int i,j;
    for(i=0;i<N;i++){
        for(j=0;j<N;j++){
            matris[i][j] = 0;
        }
    }
}
```