BLM1011 Dönem Sonu Proje Ödevi

Şevval ÇABUK

Öğrenci No: 24011107

1. **TRIVERSE NASIL OYNANIR?**

TRIVERSE, 3 oyuncu arasında oynanan bir strateji oyunudur. Tahta büyüklüğü alınan girdi doğrultusunda ayarlanır. Her oyuncu kırmızı (K), sarı (S), mavi (M) olmak üzere 3 taştan birini seçerek sırayla oynar. Oyun tahtası koyulan taşların yatay, dikey ve çaprazındaki taşların niteliğine göre sürekli güncellenir. Tahta dolduğunda en fazla taşa sahip oyuncu oyunu kazanır.

2. **ALGORİTMA MANTIĞI**

Ben projemi tasarlarken öncelikle oyuncuya oyun hakkında bilgi veren bir output yazdım. Tanımlamalar kodun okunabilirliğini etkilememesi açısından *main()* fonksiyonundan hemen sonra yazıldı.

```c
#include <stdio.h>

int main() {
    int n, i, j, x, y, strike = 0, tmpx, tmpy;
    char player[3] = {'K', 'S', 'M'};
    int win[3] = {0};

    printf("\n\n\t\tTRIVERSE\n\nGAME RULES:\n");
    printf("1) ALL COORDINATES MUST BE GIVEN IN (x,y) FORMAT.\n");
    printf("2) MAX GAME TABLE SIZE IS 23.\n");
    printf("3) FIRST PLAYER SHOULD PLACE THEIR LETTER CLOSEST TO THE MIDDLE OF THE GAME TABLE.\n\n");
    printf("4) ALL OF THE INDEX ARE STARTS FROM 0 TO N-1. PAY ATTENTION THAT YOUR COORDINATES FITS THAT STANDARD.\n");
```

n : Oyun tahtasının büyüklüğünü tutan değişken.
i, j : f : for() içerisinde dönen indisler.
x, y : Oyuncudan alınan koordinatları tutan değişken.
tmpx, tmpy : Alınan koordinatları kaybetmemek için kullanılan temporary değişken.
strike : Oyunun kaç kez döndüğünü sayan counter değişken.

Oyun tahtasını uygun büyüklükte olacak şekilde ayarladım. Geçersiz büyüklüklerde oyuncudan geçerli bir girdi verilene kadar devam eden bir *while()* oluşturdum.

Alınan oyun tahtası büyüklüğünde bir *char* array oluşturarak array elemanlarına tek tek '_' karakterini atayacak bir *for()* dizisi oluşturdum.

```
13
14      printf("ENTER NXN GAME TABLE SIZE:\n");
15      scanf("%d", &n);
16      while (n > 23 || n < 3) {
17          printf("\nSIZE MUST BE BETWEEN 3 AND 23.\nENTER AGAIN: ");
18          scanf("%d", &n);
19      } //controlling game table size
20
21      char game[n][n];
22      for (i = 0; i < n; i++) {
23          for (j = 0; j < n; j++) {
24              game[i][j] = '_';
25          }
26      }
```

OUTPUT:

```
C:\Users\sewow\Desktop\bölüm sonu projesi>24011107.exe


               TRIVERSE

GAME RULES:
1) ALL COORDINATES HAS TO BE GIVEN IN (x,y) FORMAT.
2) MAX GAME TABLE SIZE IS 23.
3) FIRST PLAYER SHOULD PLACE THEIR LETTER CLOSEST TO THE MIDDLE OF THE GAME TABLE.


ENTER NXN GAME TABLE SIZE:
```

Oyunu Döndüren Ana *while()* Döngüsü:

strike değişkeni yanlızca oyuncular geçerli input girdiğinde değiştirilmeliydi, dolayısıyla oyun belirli bir sayıda dönmeyecekti. Oyuncuların yanlış input vermesi olası olduğu için oyunu döndürecek ana döngüyü *while()* kullanarak yazdım. Oyunun yapısı gereği *strike* değişkenini yanlızca doğru input girildiğinde arttırdığımdan döngüde condition olarak tahta büyüklüğünü girdim.

Diziyi oyuncuların koordinatları daha rahat girebilmesi açısından tahtanın etrafına indisleri çerçeve şeklinde yerleştirecek şekilde yazdıran bir *for()* dizisi oluşturdum. Daha sonra

kullanıcıdan input alması için bir kısım oluşturdum. Hangi kullanıcının oynaması gerektiğini gösterebilmek için *strike* değerini kullandım.

```
27
28      while (strike<n*n) {
29          for (i=0;i<=n;i++) {
30              if (i==0) {
31                  printf("   ");
32                  for (j=0;j<n;j++) {
33                      printf("%2d ", j);
34                  }
35                  printf("\n");
36              } else {
37                  printf("%2d ",(i-1));
38                  for (j=0;j<n;j++) {
39                      printf("%2c ",game[i-1][j]);
40                  }
41                  printf("\n");
42              }
43          }
44
45          printf("PLAYER %c IS PLAYING.\nENTER COORDINATES (x,y): ", player[strike % 3]);
46          scanf("%d,%d", &x, &y);
```

OUTPUT: (n=10 için)

```
ENTER NXN GAME TABLE SIZE:
10


    0  1  2  3  4  5  6  7  8  9
 0  _  _  _  _  _  _  _  _  _  _
 1  _  _  _  _  _  _  _  _  _  _
 2  _  _  _  _  _  _  _  _  _  _
 3  _  _  _  _  _  _  _  _  _  _
 4  _  _  _  _  _  _  _  _  _  _
 5  _  _  _  _  _  _  _  _  _  _
 6  _  _  _  _  _  _  _  _  _  _
 7  _  _  _  _  _  _  _  _  _  _
 8  _  _  _  _  _  _  _  _  _  _
 9  _  _  _  _  _  _  _  _  _  _

PLAYER K IS PLAYING.
ENTER COORDINATES (x,y):
```

Input Kontrolü:

Oyunun başlangıcında *(strike=0)* koyulan taşın ortada/ortaya en yakın hücrede olması gerektiği için eğer bu şartları sağlamıyorsa kullanıcıdan tekrar input alacak bir *while()* döngüsü oluşturdum. (satır 49daki *while döngüsü*.)

Koordinatları alınan noktanın taş kümesinin etrafında olup olmadığını kontrol etmesi için bir *while()* oluşturarak alınan hücrenin etrafındaki hücrelerin ve girilen hücrenin boş olup olmadığını kontrol eden conditionlar girdim.

```
49      while (strike == 0 &&
50          ((n % 2 == 1 && !(x == n / 2 && y == n / 2)) ||
51           (n % 2 == 0 && !(x >= n / 2 - 1 && x < n / 2 + 1 &&
52           y >= n / 2 - 1 && y < n / 2 + 1)))) {
53          printf("INVALID COORDINATES. ENTER AGAIN: ");
54          scanf("%d,%d", &x, &y);
55      }
56
57      while (strike != 0 &&
58          (x < 0 || x >= n || y < 0 || y >= n || (game[x][y]!='_' ||
59          ((!( x!=n-1 && y!=0) || (game[x+1][y-1]=='_')) &&
60          (x==n-1 || game[x+1][y]=='_') &&
61          (!(x!=n-1 && y!=n-1) || (game[x+1][y+1]=='_')) &&
62          (y==0 || (game[x][y-1]=='_')) &&
63          (y==n-1 || (game[x][y+1]=='_')) &&
64          (!(x!=0 && y!=0) || (game[x-1][y-1]=='_')) &&
65          (x==0 || game[x-1][y]=='_') &&
66          (!( x!=0  && y!=n-1) || (game[x-1][y+1]=='_'))))){
67          printf("INVALID COORDINATES. ENTER AGAIN: ");
68          scanf("%d,%d", &x,&y);
69      }
```

Conditionlar (girilen koordinatlar (x,y) için):

- *strike==0 :* Girilen hamle ilk hamle değilse.
- Satır 58 : Koordinatlar tahta dışındaysa ve girilen koordinatlardaki hücre boş değilse.
- Satır 59 : Sol alttaki hücre boşsa &&,
- Satır 60 : Alt hücre boşsa &&
- Satır 61: Sağ alttaki hücre boşsa &&
- Satır 62 : Soldaki hücre boşsa &&
- Satır 63 : Sağdaki hücre boşsa &&
- Satır 64 : Sol üstteki hücre boşsa &&
- Satır 65 : Üstteki hücre boşsa &&
- Satır 66 : Sağ üstteki hücre boşsa,
  while() tekrar girdi isteyecek.

  Bu döngülerden geçen her koordinat geçerli bir koordinat olacağı için o hücreyi oynayan oyuncunun taşına eşitleyip *strike* değişkenini 1 arttırdım.

```
70
71          game[x][y] = player[strike % 3];
72          strike++;
73
```

Kontrol ve Dönüştürme:

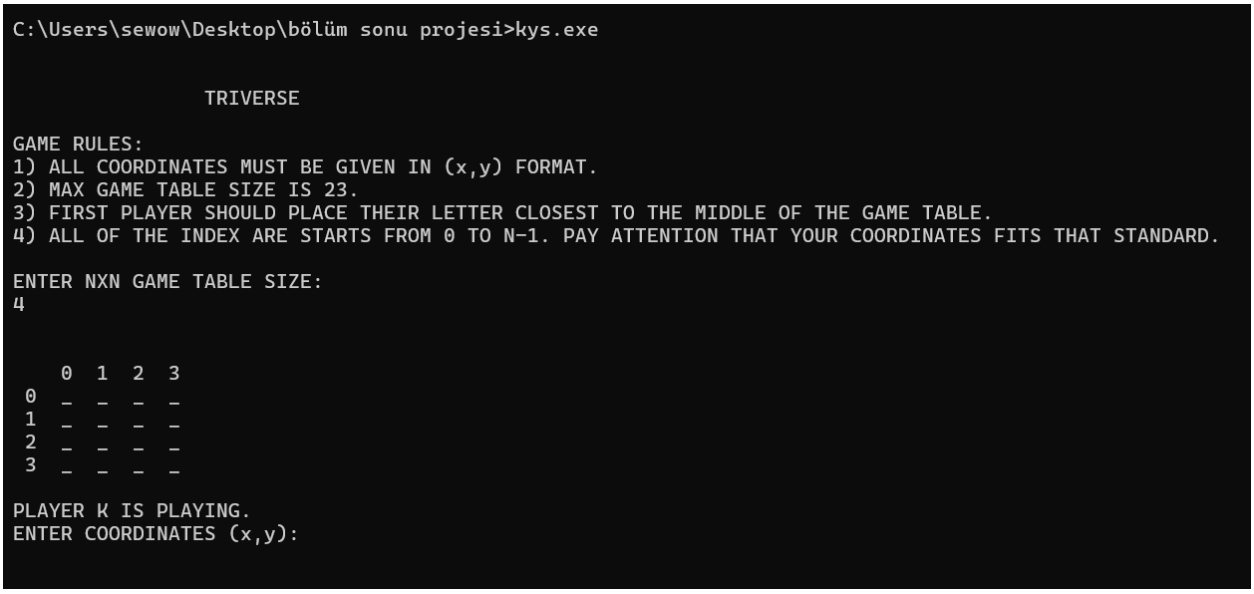
Açıklayıcı olması açısından koordinatların sağını nasıl kontrol ettiğimi anlatacağım. Çünkü hepsini aynı mantıkla yazdım.

```
74      tmpy = y + 1;
75      while (tmpy < n && game[x][tmpy] != '_') {
76          if (game[x][tmpy] == game[x][y]) {
77              for (i = y; i <= tmpy; i++) {
78                  game[x][i] = game[x][y];
79              }
80              tmpy = n;
81          }
82          tmpy++;
83      } //right
```

Sağ kontrolünün döngüsü

Sağa doğru ilerlerken asıl koordinatları kaybetmemek için (x,y) yerine (tmpx, tmpy) değişkenlerine değerleri atayıp kullandım. Ne zaman aynı taştan bulacağımı bilmediğim için *while()* kullandım ve burada temporary değişkenleri 1 arttırdım. Döngü içerisinde gezinirken her seferinde bulunduğum hücrenin değeriyle asıl koordinatlarımdaki hücrenin değerini karşılaştırdım. Eğer aynıysa bir *for()* döngüsüne sokarak geldiğim yoldan asıl koordinatlara doğru geri döndüm. Dönerken de gezdiğim hücrelerin değerlerini oyuncunun taşına eşitledim. Asıl koordinata geri döndüğümde döngüden çıkabilmesi adına temporary koordinatları sınır dışına çıkartarak döngüden çıkmasını sağladım.

3. Oyun İçerisinden Screenshotlar

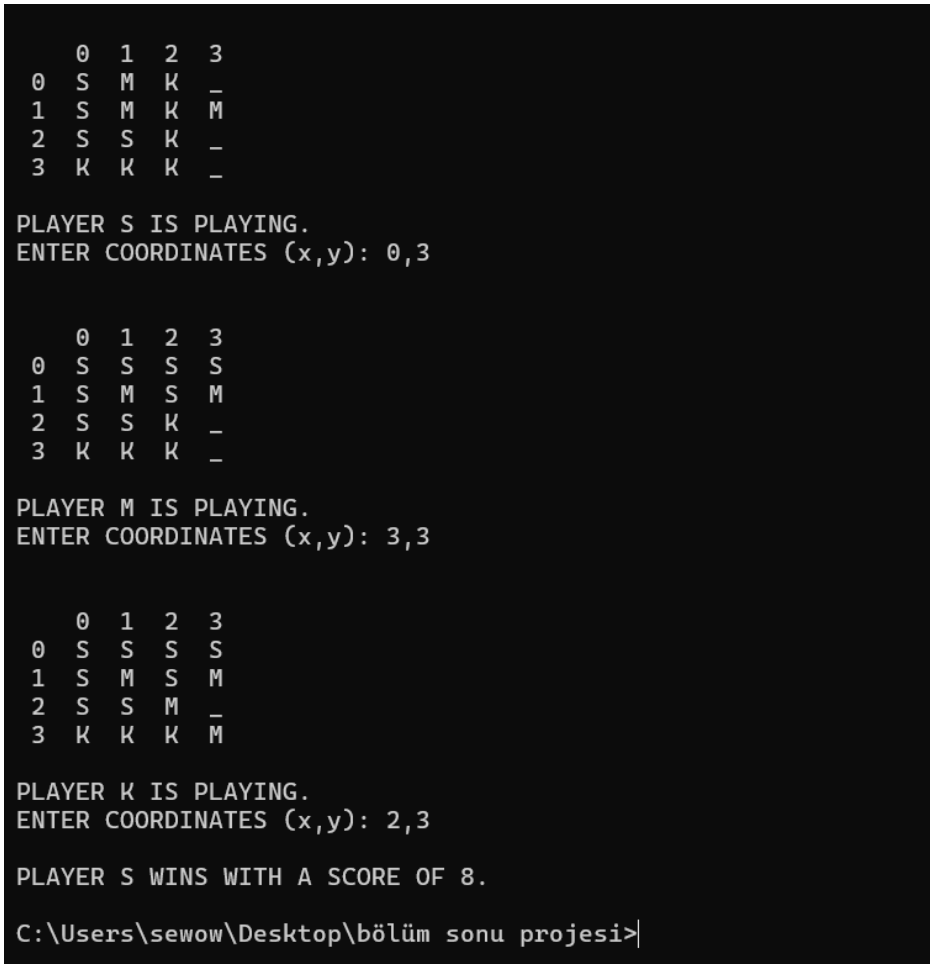


İlk koordinat alımı

```
   0  1  2  3
 0  _  _  _  _
 1  _  _  K  _
 2  _  _  _  _
 3  _  _  _  _

PLAYER S IS PLAYING.
ENTER COORDINATES (x,y): 1,1


   0  1  2  3
 0  _  _  _  _
 1  _  S  K  _
 2  _  _  _  _
 3  _  _  _  _

PLAYER M IS PLAYING.
ENTER COORDINATES (x,y): 0,1


   0  1  2  3
 0  _  M  _  _
 1  _  S  K  _
 2  _  _  _  _
 3  _  _  _  _

PLAYER K IS PLAYING.
ENTER COORDINATES (x,y): 1,0


   0  1  2  3
 0  _  M  _  _
 1  K  K  K  _
 2  _  _  _  _
 3  _  _  _  _

PLAYER S IS PLAYING.
ENTER COORDINATES (x,y): 0,0
```

```
    0  1  2  3
 0  S  M  _  _
 1  K  K  K  _
 2  _  _  _  _
 3  _  _  _  _

PLAYER M IS PLAYING.
ENTER COORDINATES (x,y): 2,1


    0  1  2  3
 0  S  M  _  _
 1  K  M  K  _
 2  _  M  _  _
 3  _  _  _  _

PLAYER K IS PLAYING.
ENTER COORDINATES (x,y): 3,2


    0  1  2  3
 0  S  M  _  _
 1  K  M  K  _
 2  _  K  _  _
 3  _  _  K  _

PLAYER S IS PLAYING.
ENTER COORDINATES (x,y): 2,2


    0  1  2  3
 0  S  M  _  _
 1  K  S  K  _
 2  _  K  S  _
 3  _  _  K  _

PLAYER M IS PLAYING.
ENTER COORDINATES (x,y): 3,1
```

```
    0  1  2  3
 0  S  M  _  _
 1  K  M  K  _
 2  _  M  S  _
 3  _  M  K  _

PLAYER K IS PLAYING.
ENTER COORDINATES (x,y): 3,0


    0  1  2  3
 0  S  M  _  _
 1  K  M  K  _
 2  _  K  S  _
 3  K  K  K  _

PLAYER S IS PLAYING.
ENTER COORDINATES (x,y): 2,0


    0  1  2  3
 0  S  M  _  _
 1  S  M  K  _
 2  S  S  S  _
 3  K  K  K  _

PLAYER M IS PLAYING.
ENTER COORDINATES (x,y): 1,3


    0  1  2  3
 0  S  M  _  _
 1  S  M  M  M
 2  S  S  S  _
 3  K  K  K  _

PLAYER K IS PLAYING.
ENTER COORDINATES (x,y): 0,2
```



Projemi açıkladığım video linki:

https://youtu.be/rbxr2IwGGic