**Gazi Üniversitesi**

**Teknoloji Fakültesi**

**Bilgisiyar Mühendisliği**

**BMT-103 Algoritmalar ve Programlamaya Giriş Dersi**

**Dönem Sonu Projesi: C Programlama Dilinde Satranç**

**Şube-2**

**23181616020 Sinan Karpuz**

**23181617002 Yusuf Buruç Adakul**

**23181617004 Sefer Ağca**

**Satranç ve Tarihi:**

Satranç, iki oyuncu arasında satranç tahtası ve taşları ile oynanan bir masa oyunudur. Bu oyun satranç tahtası denilen 8×8'lik kare bir alan üzerinde 32 adet satranç taşıyla oynanır. Toplam 64 karenin yarısı siyah, yarısı beyaz renklerden oluşur. Taraflar beyaz ve siyah renkli taşları alır, her oyuncunun bir seferde bir hamle yapmasıyla oyun gelişir. Oyunun başında beyaz ve siyahların 16 taşı bulunur. Bunlar bir şah, bir vezir, iki kale, iki fil, iki at ve sekiz piyondan oluşur.

Oyunun amacı karşı tarafın şahını mat etmektir.

MÖ 2000'li yıllarda satrancın oynandığına dair bulgular Mısır'da piramitlerdeki kabartmalarda vardır.

Satranç, kesinlik olarak ise MS 6. yüzyılda Hindistan'da ortaya çıktı. MS 10. yüzyıla gelindiğinde tüm Asya'ya, Ortadoğu ve Avrupa'ya yayılmıştı. En geç 15. yüzyıldan itibaren Avrupa'da soylular arasında çok popüler bir oyun haline geldiğinden "kraliyet oyunu" olarak anılmaya başlandı.

Kurallar ve dizilişler zaman içerisinde çeşitli değişiklikler gösterdi ve 19. yüzyılda bugünkü standart halini aldı. 20. yüzyıl Avrupası'nda toplumun entelektüel üst tabakaları arasında yayıldı ve dünyanın en popüler oyunlarından biri haline geldi.

**Programın İç Yapısı:**

C dilinde satranç yapabilmek için satranç tahtası, satranç taşlarını tanımlamak, taşların hareketlerini tanımlamak, oyuncuların geçerli hamle yapıp yapmamasını kontrol etmek ve oyun sonunun algoritması gibi birçok alt başlık incelememiz gerekiyor.

**SATRANÇ TAHTASININ TEMSİL EDİLMESİ(OLUŞTURULMASI):**

Öncelikle satranç tahtasından başlayalım. Satranç tahtasını programımız için 9x9 luk bir dizi şeklinde oluşturduk ve bu dizimizi bir fonksiyon olarak tanımladık. Beyaz taşlar yerine büyük harfler; siyah taşlar yerine de küçük harfler tanımladık .Dizimizi fonksiyon olarak tanımlamamızın nedeni her hamle sonunda elimizde yeni bir satranç tahtası oluşması sonucu ekrana çıktı verirken karışıklık olmamasıydı.

elektronik donanım, ekran görüntüsü, giriş cihazı, bilgisayar içeren bir resim

Açıklama otomatik olarak oluşturuldu

elektronik donanım, ekran görüntüsü, klavye, giriş cihazı içeren bir resim

Açıklama otomatik olarak oluşturuldu

Oluşturduğumuz sanal tahtamız. (tahta2)

Elimizde 2 tane satranç tahtası vardır. Bunun sebebi taş hareketleridir. Birinci tahta ekrana verilen ve değişiklik olduğunda bunu ekrana yansıtan tahtadır. Sanal(tahta2) tahtamız ise birinci tahtanın düzgün işleyebilmesi için gerekli olan tahtadır. Detayına inersek taşların kendilerine özgü hareket kabiliyetleri vardır. Buna bağlı olarak taşların birbirleri ile bağlantısı(taş yeme, kendi taşını yiyememe, geçersiz hamle, geçerli hamlelerin gösterilmesi) için sanal tahta ekrana verilen tahta ile entegre bir şekilde çalışır.

**OYUNUN BAŞLAMASI VE OYUNUN DEVAM ETMESİ:**

Programımızı yazarken oynamak istediğimiz taş ve nereye oynamak istediğimizi belirtmek için satır sütun numaraları belirledik.

Oyunumuza başlarken beyaz taş olarak büyük harfler kullanıldığı için büyük harfler oyuna başlar. Büyük harflerin başlangıç hamlesi yapmak için oynayacağı taşın satır-sütun numarası alınır. Eğer oyuncu oynayabileceği bir satranç taşını seçtiyse( satır sütun seçtiği için küçük taşları seçebilme ihtimali de var, eğer küçük taş seçerse tekrar taş seçmesini oyuncudan ister) taşı oynamak istediği konumun satır ve sütun numarasını sırasıyla ekrana girer. Eğer ekrana girdiği satır ve sütun da seçilen taşın hareket kabileyitine uygun olarak hamle yapabileceği bir yer (örneğin at L şeklinde hamle yapabilir ama atı hamle yapamayacağı farklı bir satır sütuna tanımlarsa kullanıcıdan tekrar işlem yapmasını ister) ise hamle programsal olarak başarılı bir hamle sayılır ve sıra küçük taşlara geçer.

**HAMLE SIRASI:**

Sıranın küçük taşlara geçmesi ise if komutu ile sağlanır. Eğer büyük taşlar başlangıç hamlesini başarılı bir şekilde yaparsa hamle sırası değişkeni if komutunun içindeki blokta 1 değerine eşitlenir. Sıra değişkeni 0’a eşitse hamle sırası büyük taşların, 1’e eşitse de hamle sırası küçük taşların olur. Bu yüzden kullanıcılar her geçerli hamle yaptığında bir sonraki kullanıcı için değeri 0 ya da 1’e eşitler.

Küçük taşlar hamle yaptıktan sonraki hamle sırasının büyük taşlara geçmesi için hamle sırası değişkeninin sıfıra eşitlenmesi sonucu sıranın büyüklere gelmesi

**metin, ekran görüntüsü, yazı tipi içeren bir resim

Açıklama otomatik olarak oluşturuldu**

**TAŞLARIN HAREKETİ:**

Satranç tahtasını ve taşların hareket prensiplerini (satır-sütun işlemleri) programa döktükten sonra oyunun işleyişi için taşların kendilerine özgü hareketlerini ayrı ayrı incelememiz gerekti. Elimizde 6 çeşit, kendine özgü hareket kabiliyeti olan satranç taşları var. Bu durum için switch-case kullanarak her taş için case açtık. Dizimizde de satranç taşlarımız ‘P’ , ‘V’ şeklinde tanımlı olduğu için case numaraları için direkt case ‘P’, case’V’ şeklinde case’ler açabildik. Daha sonrasında her taşın case’i içine taşın hareket kabiliyetine göre koşullar yazdık.

metin, ekran görüntüsü, yazılım, multimedya yazılımı içeren bir resim

Açıklama otomatik olarak oluşturuldu

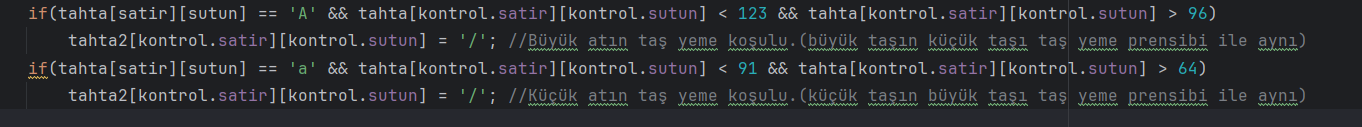
At için oluşturduğumuz case durumu. At L şeklinde hareket kabiliyetine sahip olduğu için 8 farklı şekilde hareket edebilir. 8 durum üstte oynanmak istenilen hamleye göre tanımlanmıştır.

**TAŞLARIN BİRBİRİNİ YEMESİ:**

Taşların hareketlerini tek tek programa işledikten sonra taşların birbirlerine göre durumlarını ele aldık. Örneğin taşların birbirlerini yeme durumu:

Büyük taşlar küçük taşları, küçük taşlar ise büyük taşları yiyebilir. Bunun aksi durumunda hamle geçersiz hamle olur. Bu koşula uyabilmek ve programa işleyebilmek için ASCII tablosunu kullandık. Hatırlatma amaçlı baştaki satranç tahtamızda dizilerimizin elemanları ‘P’,’V’,’v’ gibi karakterlerdir. Büyük harfleri ve küçük harfleri siyah-beyaz taşlar yerine kullanmamızın sebeplerinden biri de taş yeme hamlesi için ASCII tablosundan faydalanabilmektir.

Büyük harflerin ASCII tablosundaki değer aralığı 65’ten 90’a kadarken küçük harflerin değer karşılığı ise 97’den 122’ ye kadardır. İşte bu farktan dolayı taşlar birbirini yiyebilmektedir. Örnek üzerinde anlatacak olursak:



Büyük atın ve küçük atın taşları yiyebilme hamlesinin(diğer taşlar için de aynı yöntem geçerli) nasıl gerçekleştiği hakkındaki program kodu.

Açıklayacak olursak büyük at oynanmak istenen konuma girildiğinde eğer orada bir taş ile karşılaşırsa:

Büyük taş ile karşılaşırsa: karşılaştığı taş 65 ile 90 aralığında olduğu için (kendi taşı) o taşı yiyemez ve ekrana geçersiz hamle tekrar taş seçiniz yazısı gelir.

Küçük taş ile karşılaşırsa: karşılaştığı tas 97 ile 122 aralığında olduğu için (rakip taş) hamle yaptığı satır sütundaki bulunan taşı yiyebilir. (Rakip taş yerine hamle yapılan taş atandığından dolayı taş ekranda yenilmiş olarak gözükür.)

Bütün taşlar için taş yeme kuralı ASCII tablosundaki bu değerlere göre çalışır. Kısacası büyük taşlar ve küçük taşların ASCII tablosundaki değer farklılıklarını kullanarak taşların birbirini yiyebilmesi ve hamle yapabilmesi sorununu da çözmüş olduk.

**ŞAH-MAT OLAYI**:

Satranç tahtasını, satranç taşları hareketlerini, birbiri ile etkileşimlerini (taş yeme) ve oyuncu sırası maddelerini çözüme kavuşturduktan sonra elimizde oyunun sonlanacağı bir program olabilmesi için şah hareketlerini tekrar inceledik. Grupça tartışmamız sonucu çözüm yolu olarak bu yöntemi programımıza işledik.

Örneğin büyük harfleri kullanan oyuncu küçük taşlardaki oyuncuyu yaptığı hamle ile şah-mat etti. Sıra küçük taşlara geldiğinde oyuncunun hamle yapabilecek bir yeri olmadığından dolayı mat rotasındaki bir yere hamle yapar. Küçükler hamle yaptıktan sonra sıra büyüklere geçeceği için büyük taşlardaki matı yapan taş küçük şahı yer ve oyun sonlanır. Buradaki koşul ise şudur:

Şah-mat yapan oyuncunun oyunu bitirmesi için şah çektiği taşın satır sütun numarasını girip şah çektiği taşını seçtikten sonra, oynamak istediği satır ve sütun numaraları yerine de rakibinin şahının bulunduğu satır ve sütun numaralarını girerse oyunu kazanmış olur.

metin, yazı tipi, ekran görüntüsü içeren bir resim

Açıklama otomatik olarak oluşturuldu

**FONKSİYONLAR VE İŞLEVLERİ**:

tahtayiGoster

Bu fonksiyon tahtanın en güncel halini ekrana çıktı olarak verir(printler).

İçinde for döngüsü kullanarak tahtanın güncellenmiş halini sırasıyla 1. Satırdaki sütunlar, 2. Satırdaki sütunlar …… şeklinde yazdırır. Böylece tahtanın güncel hali ekrana yansımış olur.

**GecerliHamleler**

Bütün taş hamleleri içinde switch-case yapısıyla bulunduran fonksiyondur. Taş hareketleri, taş yeme ve diğer taş hamleleri tek tek case’ler ile bu fonksiyonun altına yazılmıştır. Sanal tahtanın önemi ise bu fonksiyonda belirginleşir.

Örneğin büyük fil (3. Satır 3. Sütun) küçük bir piyonu(5. Satır 5. Sütun)yeme durumunda olsun. Filin hareketi satranç kurallarına göre çaprazdır ve önüne taş çıkmadığı sürece çapraz gidebilir. Önüne taş çıktığındaysa iki durum vardır:

1. **DURUM:**

Kendi taşı olma durumunda istenilen yere oynayamaz. Programlamada hata oluşmaması için kendi taşından sonraki çapraz hamlesine oynamasına izin verilmez(çapraz gitmesini durdurur). Aksi takdirde normal satrançta geçerli olmayan bir hamle konum ataması yaptığımız için direkt oraya oynar ve bu yanlış bir hamle olmasına rağmen program çalışmaya devam ederdi. Bu yüzden sanal tahta(tahta2) bu hareket hamlelerini kısıtlamak için bulunur. Bu hamle bizim programımızda da geçersiz hamle olup kullanıcıdan tekrar taş seçilmesini talep eder.

Örneğin 3. Satır 3. Sütundaki fili 5. Satır 5. Sütuna oynamak isteyelim. Fakat 4. Satır 4. Sütunda kendi taşımız olsun. Satrançta 4. Satır 4. Sütunu atlayamayacağımız için bizim programımızda da atlamaması gerek. Eğer elimizde ikinci bir tahta olmasaydı bu kuralı gerçekleştiremezdik fakat geçerli hamleler fonksiyonundaki caselerin (taşlar için geçerli hamleler) içindeki tahta2 komutları ile beraber bu kuralı uygulayabiliyoruz.

1. DURUM:

Rakip taş olma durumunda ( rakip taşın önünde kendi taşı olmadığı sürece) ise oradaki taşı yiyebilir. Kod hata vermeden ve başarılı bir şekilde çalışmaya devam eder.

**tahtayiDuzelt**

Girilen satır sütundaki taşın (seçilen taş) geçerli hamle olarak oynayabileceği yerler \* ile gösterilir.

tahtayiDuzelt fonksiyonu ekrandaki oynanabilen hamleler gösterildikten sonra tahta çıktısındaki "\*" ları tekrar "-" karakterine dönüştüren bir fonksiyondur.

metin, ekran görüntüsü, yazı tipi, tipografi içeren bir resim

Açıklama otomatik olarak oluşturuldu

Bu çıktıda görüldüğü üzere büyük taşlar taş seçiminde 7. Satır 5. Sütundaki taşı seçmiş. Bu hamle seçilen piyonun ilk hamlesi olduğu için piyona 2 hamle hakkı tanınmış. \* bulunan satırlar belirtilen piyonun hamle yapabileceği yerleri gösterir. Yani bu piyon bu durumda 6. Satır 5. Sütun ve 5. Satır 5. Sütun bölgelerine oynayabilir.

Oyuncu hamlesini yaptıktan sonra diğer oyuncu için tahtanın güncel hali ekrana gelmelidir. Fakat bu güncel halde sadece taş değişikleri istenildiği için geçerli hamleler için koyulan \* işareti tekrar – işaretine dönmelidir. tahtayiDuzelt fonksiyonu da bu işlevi yerine getirir.

**tasiOyna**

Geçerli hamleler fonksiyonunda taşın oynayabileceği satır sütunlara \* işareti koyulur. tasiOyna fonksiyonu ise girilen satır sütun hamlesine seçilen (satır sütun numarası girilen taş) taşı atar ve taş oraya hareket eder. Bunu yaparken de oynanan taşın yeri normal satrançta boş kalacağı için bizim programımızda boşluk anlamına gelen – işaretini taş seçimi için istenen satır sütuna atar.

Örneğin satır 2 sütun 1’deki küçük piyonu satır 4 sütun 1’e oynadığımda oradaki – işareti yerine piyonu atar. Piyon oraya atanacağından yerinin boş kalmaması için piyonun eski konumuna da – işaretini koyar.

**AKIŞ ŞEMASI:**

