**LABİRENT OYUNU**

1. Ogulnabat HANGELDİYEVA 2. Sema DEMİR

Bilgisayar Mühendisliği Bölümü

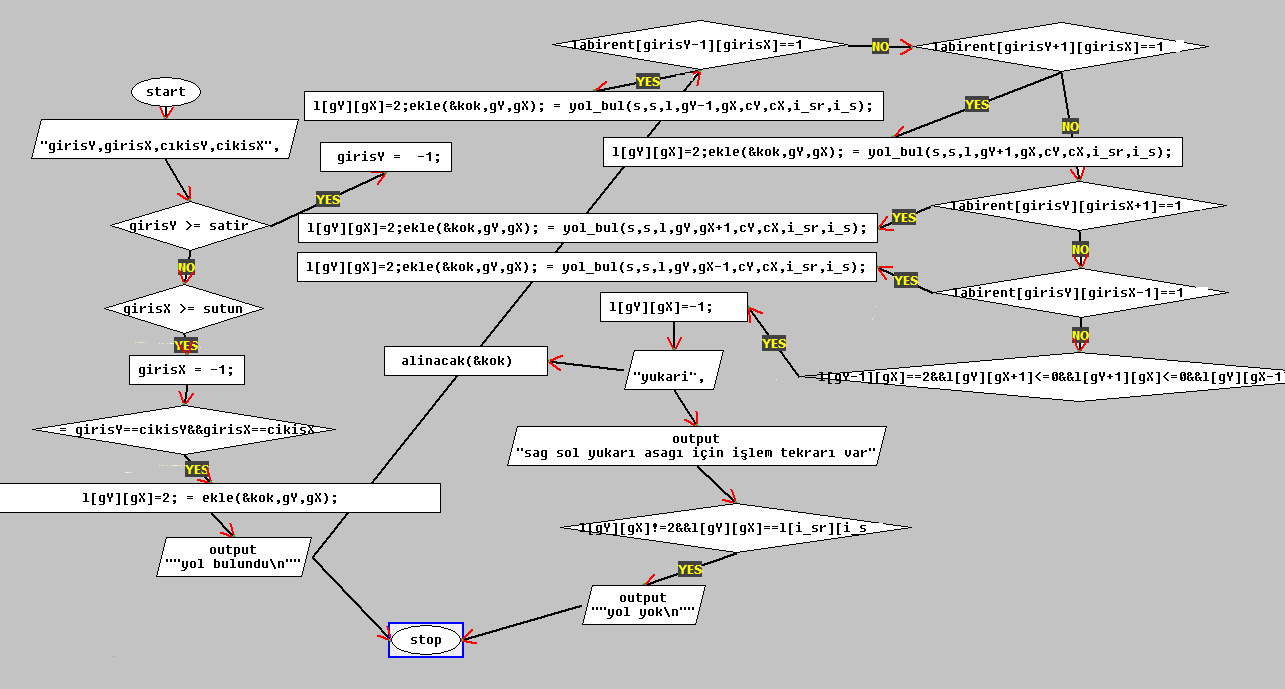
Kocaeli Üniversitesi

1. [nabatmn@gmail.com](mailto:nabatmn@gmail.com) 2. [sema.demir.1997@gmail.com](mailto:sema.demir.1997@gmail.com)

**Projenin Özeti**

Projenin amacı öğrencilere bağlı liste ve yığın mantığını öğretmek amaçlanmıştır. Yığın yani Stack yapısı son giren ilk çıkar (LIFO ) mantığında çalışır yani labirentte kullanmamızın temel nedeni yol varsa yığına atmamız yol olmadığı durumda ise son bulunan yoldan başlayarak yığından

çıkarılması işlemidir. Stack yapısı bir yazılımcının bilmesi gerekenler arasında yer almaktadır. Stack yapısı sadece programlama konusu olmayıp veri yapıları konusudur . Bağlı listelerle yığın yapısı işlemi zorlaştırsada bizim profesyonel yazılımcı olmamızı sağlayacaktır.



1. **Giriş**

Bize verilmiş olan Labirent Oyunu projesini hem Türkçe hem de İngilizce kaynaklarda arattık. Labirent Oyunu çoğu dilde çok kolay bir şekilde yazılan bir oyunmuş çoğu kişi labirent oyununu denemiş. Fakat bize verilen proje ile piyasada yapılmış olan projeler arasında büyük bir farklılık vardı. Farklılığın sebebi de bizim projemizde bağlı liste ve yığın yapılarının kullanılmasının istenmesiydi. Proje dizi ve matris kullanılarak çok daha kolay bir şekilde bitiyordu fakat bizim amacımız projeyi kolay bir şekilde bitirmek değil bağlı liste ve yığın yapısını öğrenmek olduğu için biraz daha zor dolayısıyla biraz daha fazla öğretici olan bir şekilde projeyi tamamlamış olduk.

Yaptığımız projenin çıktısı 4. Sonuçlar bölümünde verilmiştir.

1. **Temel Bilgiler**

Projenin yazılımında Code::Blocks un C programlama dili kullanıldı. .Kütüphane olarak #include <stdio.h> #include <stdlib.h> #include<time.h> dili kütüphaneleri kullanıldı.

1. **Diğer Bölümler**
2. **Main()**

Main() fonsiyonu içerisinde matrisin satır ve sütun değerini kullanıcıdan alır ve 0 ve 1 lerden oluşan rastgele matris oluşur.İşlemin ardından kullanıcıdan giriş kapılılarını X ve Y koordinatlaını kullanıcıdan isteriz. İşlem aynı şekilde çıkış kapısı için de yapılır .Eğer kullanıcı matisin dışında olan ya da kenarda olmayan bir hücre seçerse uyarı verir ve işlem tekrarlanır.

Eger kullanıcının girdiği hucrede duvar yani 0 değeri ver ise yine uyarı verir ve tekrar giriş ve çıkış kapılarını girmesini ister . Kullanıcı doğru kapıları seçtikten sonra yol\_bul() foksiyonu çağırılır. Yol\_bul()foksiyonun çalışmasının ardından eğer yol var ise stack yapısında bulunan yollarin indisleri ekrana yazdırılır ve bulunan yol matriste görüntülenecek şekilde ekrana

yazdırılır ve ekranda “Yol bulundu” yazısı yazdırılır.Eğer yol bulumadıysa “Ekranda yol bulunmadı “ yazısı yazdırılır.

1. **Yol\_bul()**

yol\_bul () foksiyonu parametre olarak matris,giris kapıların koordinatları ve çıkış kapılaırının koordinatlarını alınır.

yol\_bul()fonksiyonu recursive mantığında çalışır. Fonksiyon içerisinde indis matrisin dışarısına çıktığı durumlarda -1 değeri atar.Eğer giriş kapısı çıkış kapısına eşit ise yol bulunduğu anlamına gelir ve çıkış yapar bu işlemi if kontrolu ile yapılmaktadir. İf kontrolu ile bulunduğu indisinin sağ sol aşağı yukarısını kontrol eder eğer yol varsa o indise 2 değeri atanır ve ekle() fonsiyonu çağırılarak

bulunan indisi yığına aktarır ve foksiyon yine kedini çağırır,eğer yol yok ise bulundğu indise -1 değerini atanır ve alınacak() fonsiyonu çağırılarak bir önceki atanan indis değer

indisten çıkartılır . Son olarak eğer labirentin giriş kapısı 2 değerini almıyorsa ve yolun geldiği yer giriş kapısına eşit ise yol bulunmadı anlamına gelir ve işlem yol bulunmadı olarak tamamlanır.

1. **Ekle()**

Ekle() fonksiyonu ile parametre olarak 2 tane değer alır bu değerler bulunan yolların indisleridir. Aldığı veriyi yığına atama işlemi yapar.

1. **Alınacak()**

Alınacak() fonsiyonu ile parametre olarak işlenecek olan yolun indisini alır ve yığında bulunan bu indisi yığından çıkartır.

1. **Struct Yıgındugum \*kok**

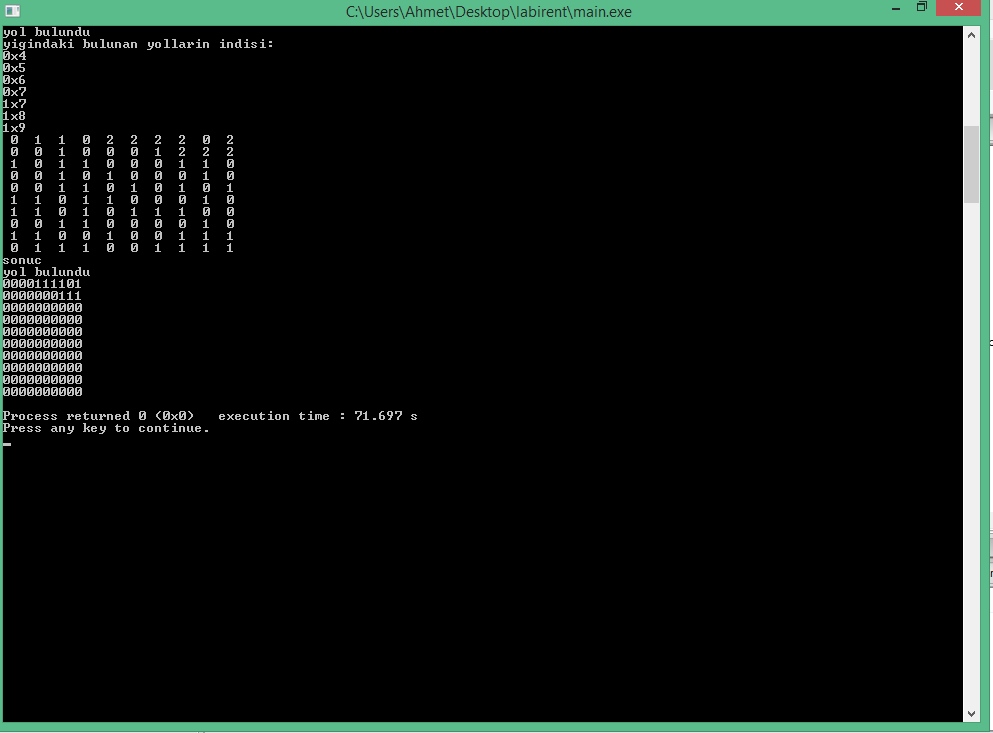
Struct kok yığında tutulacak olan değerleri barındırır. Kök yapısı bağlı liste yapısından oluşmaktadır .

Yol bulunursa main() fonksiyonu içinde çagırılır yol yok ise kök yığın içerisi boş olacaktır.

1. **Sonuçlar**

Sonuç olarak kod çalışır durumunda yol bulma işlemini gerçekleştiriyor.

Kod çalışıyor durumda bazı yöntemlerde kısa yol bulmaya çalıştık . Proje bizim açımızdan çok verimli oldu. Bağlı liste ve yığın mantığını daha iyi anlamamıza olanak sağladı. Veri yapıları bilgilerimizi yenilememizi ve pekiştrmemizi sağladı ,bu yuzden Labirent Oyunu projesinin verilmesi eksik kalan bilgilerimizi tamamlama açısından güzel düşünülmüş proje oldu. Ekran çıktısı aşağıda verilmiştir.



**5. Kaynakça**

Bizden onceki yapılmış uygulamalar

[1]https://www.youtube.com/watch?v=-BkHu8em9fM

Faydalandıgımız kaynaklar

[2]http://embedded.kocaeli.edu.tr/veriyapilari\_1718\_5/

[3]http://www.cs.bu.edu/teaching/alg/maze/?utm\_campaign=DonanimHaber&utm\_medium=referral&utm\_source=DonanimHaber

[4]https://www.youtube.com/watch?v=qMG5LvPoTis

[5]https://www.youtube.com/watch?v=0xY7GQWVAyM