**LABİRENT OYUNU**

Ali Recep KARACA, Mehmet Fırat KÖMÜRCÜ

Bilgisayar Mühendisliği Bölümü

[recepkaraca@yandex.com](mailto:recepkaraca@yandx.com) , [mehmetfiratkomurcu@hotmail.com](mailto:mehmetfiratkomurcu@hotmail.com)

**Özet**

*Bu program; nxn boyutlu, n değeri kullanıcı tarafından girilen bir matris üzerinde rastgele yol ve duvarlar oluşturmakta olup, oluşan bu rastgele yol ve duvarlara göre bağlı liste ve yığın yapısını kullanarak, kullanıcı tarafından girilmiş olan giriş ve çıkış kapıları arasında yol olup olmadığını bulmaktadır. Eğer yol bulunamaz ise kullanıcıya yolun bulunamadığına dair bilgi verilme, yol bulundu ise; bulunan yol kullanıcıya gösterilmektedir.*

**1. Problem Tanımı**

Bu projede çözümlenmesi gereken sorun; nxn boyutlu, n değeri kullanıcı tarafından girilmiş olan bir matris üzerinde, yine giriş ve çıkış kapıları kullanıcı tarafından girilmiş olan rastgele oluşturulmuş bir labirent üzerinde, giriş ve çıkış kapıları arasındaki yolun sadece bağlı liste ve yığın yapısı kullanılarak bulunmasıdır.

**2. Giriş**

Verilen projenin konusu; rastgele oluşturulan bir labirent üzerindeki, belirlenen kapılar arasındaki çıkış yolunun bulunmasıdır.

Verilen projenin amacı ise; rastgele oluşturulan labirent üzerindeki, belirlenen kapılar arasındaki çıkış yolunun bulunması problemi çözümü sırasında sadece ve sadece bağlı liste ve yığın yapısı kullanılacağından dolayı, öğrencilerin bağlı liste ve yığın yapısına odaklanmalarını sağlamak, dolayısıyla öğrencilerin bu iki konuyu daha iyi anlamalarını ve bunun üzerinde pekiştirme yapmalarını sağlamaktır.

**3. Temel Bilgiler**

Projeyi gerçekleştirme aşamasında C programlama dili kullanılmıştır. Yapmış olduğumuz projede C dilinde bulunan “graphics.h” ve “time.h” gibi kütüphaneler projeye dahil edilerek, bu kütüphanelerden, bazı sorunların çözülmesi adına yardım alınmıştır.

Yazılan bu program Windows 10 üzerindeki Codeblocks derleyicisi üzerinde derlenmiş olup, derleme aşamasında herhangi bir hatayla karşılaşılmamıştır. Ayrıca derlenen bu program yine Windows 10 üzerinde çalıştırılmış ve programın çalışması sırasında yine herhangi bir hatayla karşılaşılmamıştır.

**4. Genel Yapı**

Yaptığımız projede ilk olarak; nxn boyutlu rastgele olarak oluşturulacak olan matrisin n değeri, kullanıcıdan istenmiştir. Bu işlem sırasında kullanıcının girebileceği maksimum n değeri 18 olarak belirtilmiştir. Bunun sebebi 18’den büyük matrislerin ekrana sığamamasıdır ve sığamayan matrislerin ekranda görülememesidir. Ayrıca kullanıcının harf mi sayı mı girdiği de sürekli kontrol edilmektedir, eğer sayı haricinde başka bir karakter girdiyse program kullanıcıdan tekrar tekrar n değerinin girilmesini istemektedir.

Program kullanıcıdan n değerini aldıktan sonra; “listeolustur” isimli fonksiyona bu n değerini göndererek, bu fonksiyon içerisinde kullanıcı tarafından girilmiş olan n sayısı boyutunda bir bağlı liste oluşturmakta ve bu bağlı liste içerisine rastgele olarak 0 veya 1 değerleri atamaktadır. Burada 0 duvarları, 1 ise yolları belirtmektedir.

Ardından; program kullanıcı tarafından girilmiş olan n değerini “listegoster” isimli fonksiyona göndermektedir ve bu fonksiyon içerisinde “graphics.h” kütüphanesinden yararlanarak , kullanıcıya konsol ekranında labirenti göstermek yerine ekstra bir grafik penceresinde, göze daha hoş gelecek bir tasarım içerisinde, rastgele 0 ve 1 değerlerinden oluşan labirent, ekrana 0 ve 1 yazdırılmak yerine bu pencere içerisinde renkli kutucuklara dönüştürülmüş olup, burada 0’lar kırmızı kutucuk , 1’ler ise beyaz kutucuk olacak şekilde kullanıcıya gösterilmiştir. Ayrıca yine bu fonksiyonda, giriş ve çıkış kapılarının koordinatları ekrana yazdırılarak kullanıcının tek tek koordinat hesaplama sorununun çözümlenmesi amaçlanmıştır.

Oluşturulan labirent kullanıcıya “graphics.h” kütüphanesinden yararlanılarak hoş bir pencere içerisinde gösterildikten sonra; kullanıcıdan labirent üzerindeki hangi kapılardan giriş ve çıkış yapılacağı bilgisini girmesi istenmiştir. Kullanıcı burada giriş ve çıkış kapılarının x ve y değerlerini girerek giriş ve çıkış yapılacak kapıların hangi kapılar olacağını belirlemektedir. Burada; kullanıcıdan kapıların x ve y değerleri istenmekte olduğundan, daha önce kullanıcıya gösterilen labirent üzerinde, her kapının tek tek koordinatları kullanıcıya gösterilmiş olup, kullanıcı x ve y değerlerini tek tek hesaplama zahmetinden kurtulmuştur. Yine burada; kullanıcının gireceği kapının koordinatını unutma ihtimaline karşın; giriş olarak -1 girilmesi durumunda kullanıcıya tekrardan, oluşturulan labirent gösterilmekte olup, kullanıcının koordinat bilgisini tekrar hatırlaması sağlanmaktadır. Bu kapı koordinatı girişleri sırasında yine daha önceki kullanıcıdan alınan n girişinde olduğu gibi, kullanıcının girdiği değerin harf mi sayı mı olduğu kontrol edilmekte olup, girilen değerin harf olması durumunda kullanıcıdan tekrar giriş istenmektedir. Ayrıca labirente kenarlar üzerinden giriş ve çıkış yapılacağından dolayı kullanıcıdan girilen koordinatlar kontrol edilmekte olup, girilen koordinatların kenar üzerindeki bir kapıya ait olmaması durumunda kullanıcıdan tekrar kapı koordinat bilgisi istenmektedir.

Ardından; rastgele oluşturulan bu labirent içerisinde yığın ve bağlı liste yapısı kullanılarak yol bulma işlemine geçilmiştir. Burada; oluşturulan yığın yapısı içerisine ilk olarak kullanıcının girmiş olduğu giriş kapısının x ve y koordinatları atılmıştır. Ardından; çıkış kapısına ulaşılana kadar bir while döngüsü içerisinde sürekli labirent içerisinde dolaşılma işlemi gerçekleştirilmiştir.

Burada; eğer programın şu an bulunduğu koordinat labirentin en sol kenarı değilse ve şu an ki koordinatın bir solundaki kapının değeri 0 yani kapalı değilse ve değeri 2 değilse (2 değeri o kapının daha önce dolaşılmış olduğu anlamına gelmektedir. Dolaşılmış olan 1 ler dolaşıldıktan sonra 2 değerini almaktadır ve bu da programın daha önce bu noktadan geçmiş olduğunu anlamasını sağlamaktadır.) şu an ki koordinat bir sola kaydırılmaktadır yani labirent içerisindeki bir sol kutucuğa gidilmektedir. Gidilen bu kutucuğun değeri 2 yapılarak tekrar o noktadan geçilmesi engellenmektedir. Ayrıca gidilen her noktanın koordinat bilgisi tek tek yığın yapısı içerisine aktarılmaktadır.

Eğer programın şu an bulunduğu koordinat labirentin en sağ kenarı değilse ve şu an ki koordinatın bir sağındaki kapının değeri 0 yani kapalı değilse ve değeri 2 değilse, şu an ki koordinat bir sağa kaydırılmaktadır yani labirent içerisindeki bir sağ kutucuğa gidilmektedir. Gidilen bu kutucuğun değeri 2 yapılarak tekrar o noktadan geçilmesi engellenmektedir. Ayrıca gidilen her noktanın koordinat bilgisi tek tek yığın yapısı içerisine aktarılmaktadır.

Eğer programın şu an bulunduğu koordinat labirentin en üst kenarı değilse ve şu an ki koordinatın bir üstündeki kapının değeri 0 yani kapalı değilse ve değeri 2 değilse, şu an ki koordinat bir üste kaydırılmaktadır yani labirent içerisindeki bir üst kutucuğa gidilmektedir. Gidilen bu kutucuğun değeri 2 yapılarak tekrar o noktadan geçilmesi engellenmektedir. Ayrıca gidilen her noktanın koordinat bilgisi tek tek yığın yapısı içerisine aktarılmaktadır.

Eğer programın şu an bulunduğu koordinat labirentin en alt kenarı değilse ve şu an ki koordinatın bir altındaki kapının değeri 0 yani kapalı değilse ve değeri 2 değilse, şu an ki koordinat bir alta kaydırılmaktadır yani labirent içerisindeki bir alt kutucuğa gidilmektedir. Gidilen bu kutucuğun değeri 2 yapılarak tekrar o noktadan geçilmesi engellenmektedir. Ayrıca gidilen her noktanın koordinat bilgisi tek tek yığın yapısı içerisine aktarılmaktadır.

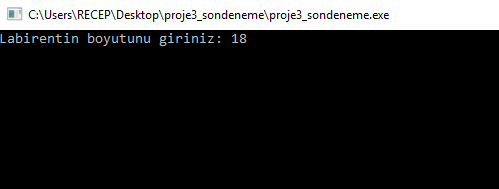
Bu tarama işlemleri sağ, sol, yukarı, aşağı yönleri için her seferinde kontrol edilmektedir. Eğer gidilecek hiçbir yön kalmadıysa yani tüm komşu hücrelerin değerleri 0 veya 2 olduysa girilen bu yol yığın yapısından çıkarılarak programın bir önceki adıma dönmesi sağlanır. Bu işlem çevre yolları açık olan herhangi bir hücreye gelene kadar tekrarlanır. Program açık olan bir yol algıladığı anda o yöne dönerek bu sefer o yöndeki bütün kutucukları tarayarak bu yollar yığın yapısına atılmaktadır ve çıkışa ulaşılmaya çalışılmaktadır. Eğer yine çıkışa ulaşılamamışsa program girilen yolları yine yığın yapısından silerek öndeki adımlara dönmektedir. Bu işlemler sonucunda eğer kullanıcının girmiş olduğu çıkış koordinatına ulaşıldıysa veya yığın yapısı içerisinde hiçbir koordinat kalmadıysa yani program labirente giriş noktasına geri döndüyse, bu durumda verilen giriş ve çıkış noktaları arasında bir yol bulunamadığı anlamına geldiğinden, program yol arama algoritmasını durdurmaktadır ve yol bulunup bulunamadığı bilgisini kullanıcıya iletmektedir. Eğer iki nokta arasında bir yol bulunabilmiş ise bu yol kullanıcıya hoş bir grafik penceresi üzerinden gösterilmektedir.

**5.Fonksiyonlar ve İşlevleri**

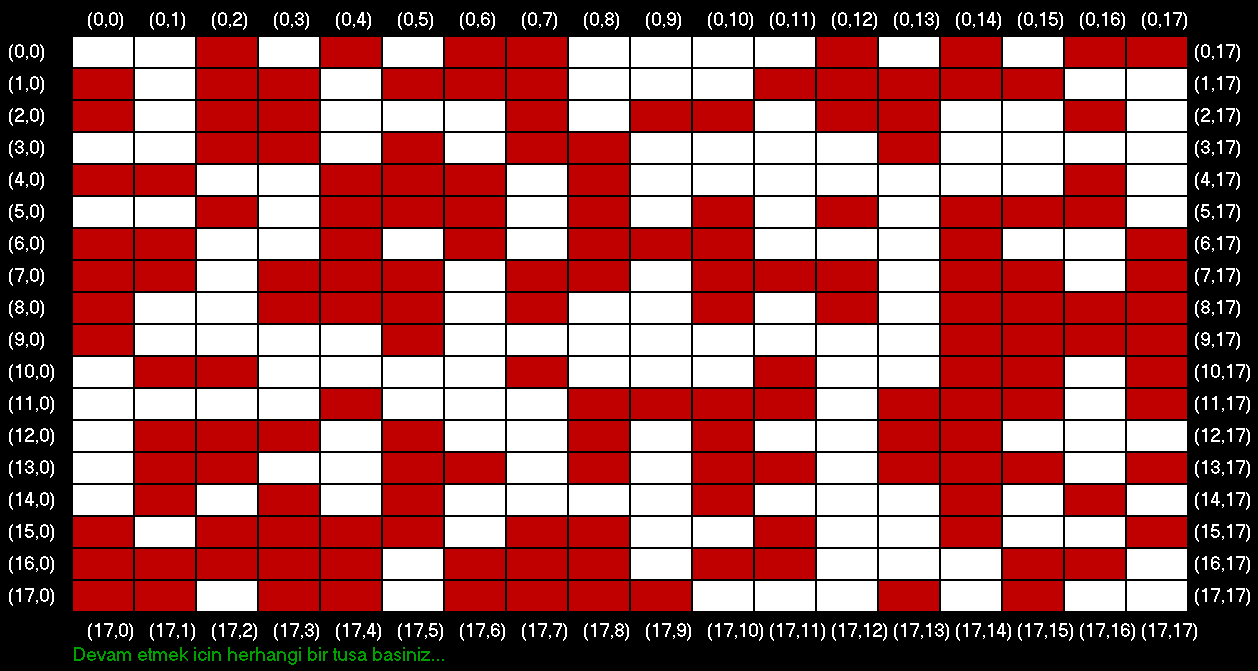
|  |  |
| --- | --- |
| **Fonksiyon** | **İşlevi** |
| bekle | Oluşturulmuş olan labirent içerisinde yol tarama işlemi yapılırken, gidilen her yolun kullanıcı tarafından görülebilmesi için, her işlem sonrasında bir miktar bekleyerek kullanıcının yapılan işlemleri algılamasının sağlanmasını sağlamak. |
| bosmu | Program içerisinde bolca yığın yapısından yararlanıldığından dolayı, program içerisinde yığından her silme işlemi yapılacağı sırada bu fonksiyon kullanılmakta olup, yığının boş olup olmadığı kontrol edilmektedir, eğer yığın yapısı boşsa silme işleminin gerçekleştirilmemesini sağlamak. |
| pop | Program içerisinde kullanılan yığın yapısı içerisinden, bosmu fonksiyonundan dönen sonuca bağlı olarak silme işlemi gerçekleştirmek, eğer bosmu fonksiyonundan bos sonucu çıkarsa bu fonksiyon çalıştırılmaz |
| push | Program içerisinde kullanılan yığın yapısı içerisine, program içerisinde belirlenen değerlerin atılmasını sağlamak. |
| yigingoster | Programdaki gidilen tüm yollar yığın içerisine atıldığından dolayı, yığının içerisinde bulunan koordinat bilgilerini ekranda göstermek. |
| listegoster | Program içerisinde rastgele matris oluşturulurken dahi dizi yapısı kullanılmadığından, oluşan değerler bağlı liste yapısı içerisinde tutulduğundan, bu bağlı liste içerisindeki matrisin, matris formunda kullanıcıya gösterilmesini sağlamak, bu gösterme işlemi sırasında kullanıcının gözüne hitap etmesi adına “graphics.h.” kütüphanesinden faydalanarak konsol ekranından bağımsız, göze hoş gözüken bir pencere içerisinde matrisi kullanıcıya göstermek. |
| listeolustur | Kullanıcı tarafından boyutu girilen labirentin boyutunu alarak, belirlenen boyut büyüklüğünde bir bağlı liste oluşturmak ve bu bağlı liste içerisine rastgele 0 ve 1 değerleri atamak.Burada 0 duvarları 1 ise yolları belirtmektedir. |
| listegosteryenilenen | Kullanıcıya, programın yol bulma işlemi yapması sırasında gidilen her yolun, “graphics.h” kütüphanesinden yararlanılarak göze hoş gelecek bir şekilde adım adım gösterilmesini sağlamak. |
| listegosterson | Programın yol bulması durumundan, kullanıcıya bulunan bu yolun “graphics.h” kütüphanesi kullanılarak gösterilmesini sağlamak. Yol bulunamaması durumunda bu fonksiyonun bir anlamı olmayacağından, bu durumda çalışmaz. |
| kapidondur | Program içerisinde sürekli, üst, alt, sağ, soldaki yoların değerlerinin okunması gerektiğinden, bu noktalar içerisinde bulunan değerlerin 0 ve 2 olması durumunda gidilmemesi, sadece yolun 1 değerinde olması durumunda o noktaya gidilmesi gerektiğinden böyle bir fonksiyona ihtiyaç duyulmuştur. Bu fonksiyon belirlenen koordinattaki noktanın içerisinde hangi değerin olduğunu döndürmektedir. |
| gitveata | Labirent üzerinde yol arama işlemi yapılırken gidilen tüm yollara 2 değeri atamaktadır, bu sayede program bir daha o noktadan geçmemektedir, boşa işlem gücü harcamamaktadır. Ayrıca bu fonksiyon bir yol tarandıktan sonra o yolun çıkmaz yol olduğu kanısına vardıktan sonra, o yola bağlı tüm noktaları kapatarak yani sıfır yaparak bir nevi duvar örme işlemi gerçekleştirmektedir. Bu işlemler bir yol ayrımına girene kadar sürmektedir. Çıkmaz yoldan çıkana kadar tüm noktalara 0 değeri ataması yapılmaktadır. |

**7.Programın Çalışma Sırasından Alınan Bazı Görüntüler**

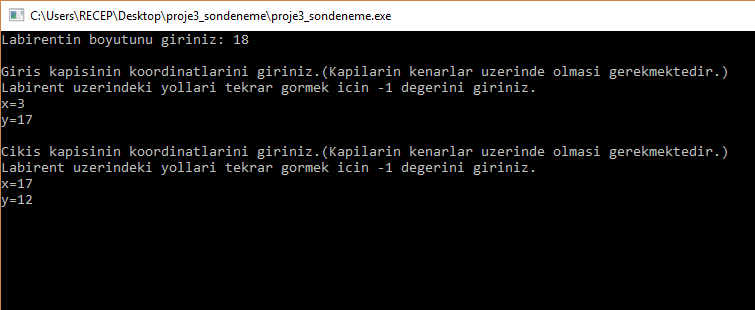
Programın giriş kısmı:



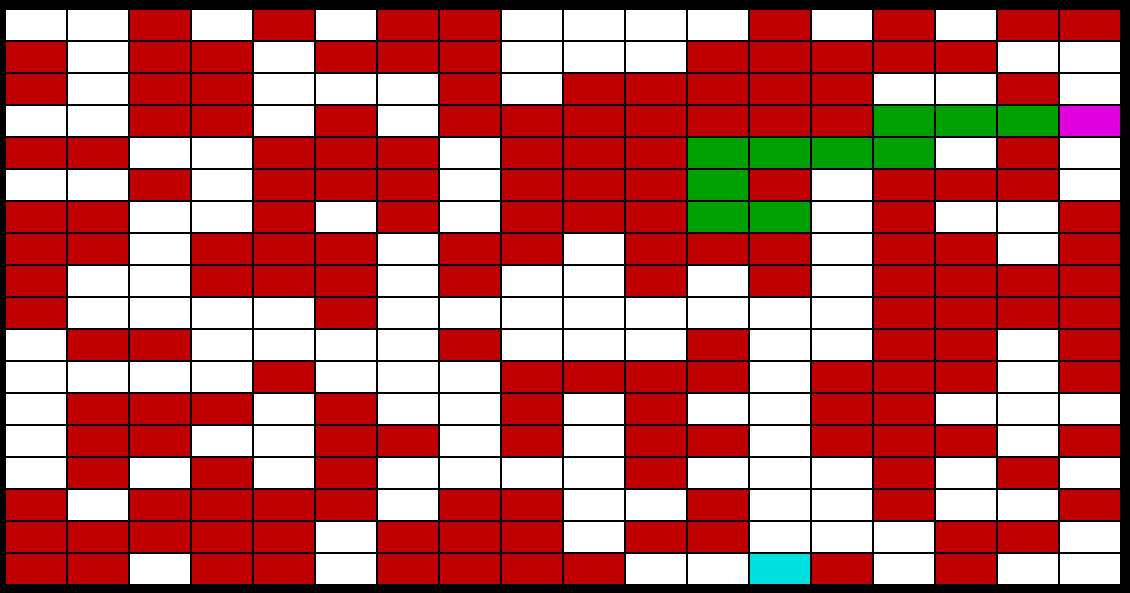
Labirentin kullanıcıya grafiksel olarak gösterilmesi:



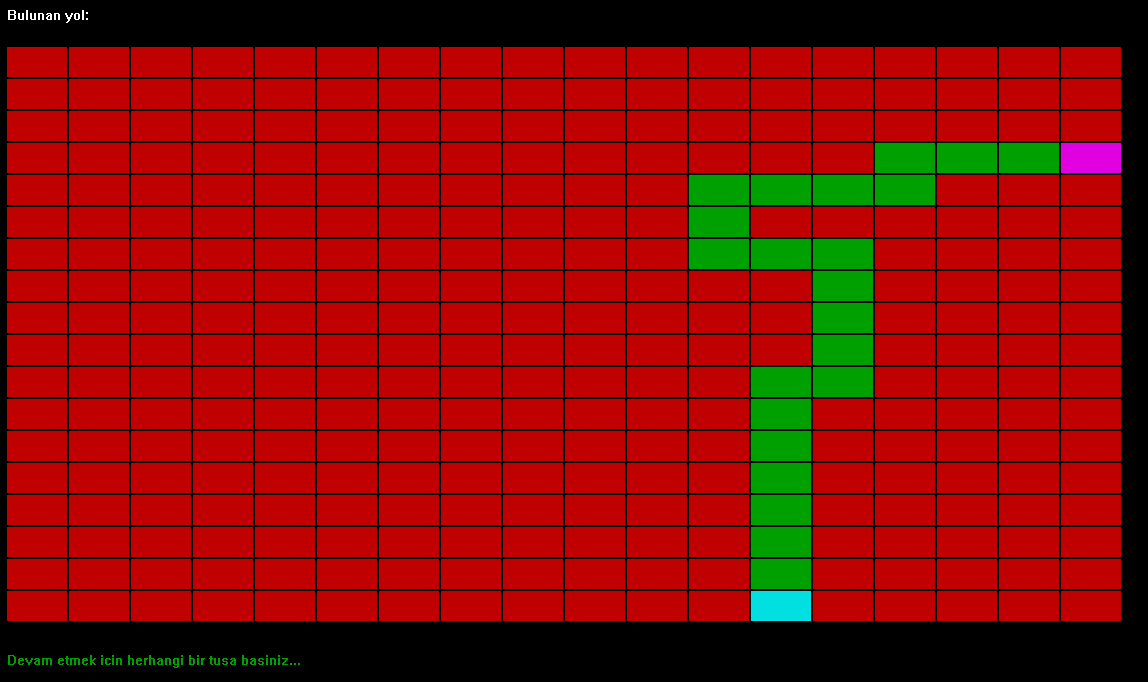
Kullanıcıdan giriş ve çıkış kapılarının alınması:



Programın yol bulma anından bir görüntü:



Bulunan yolun kullanıcıya gösterilmesi:

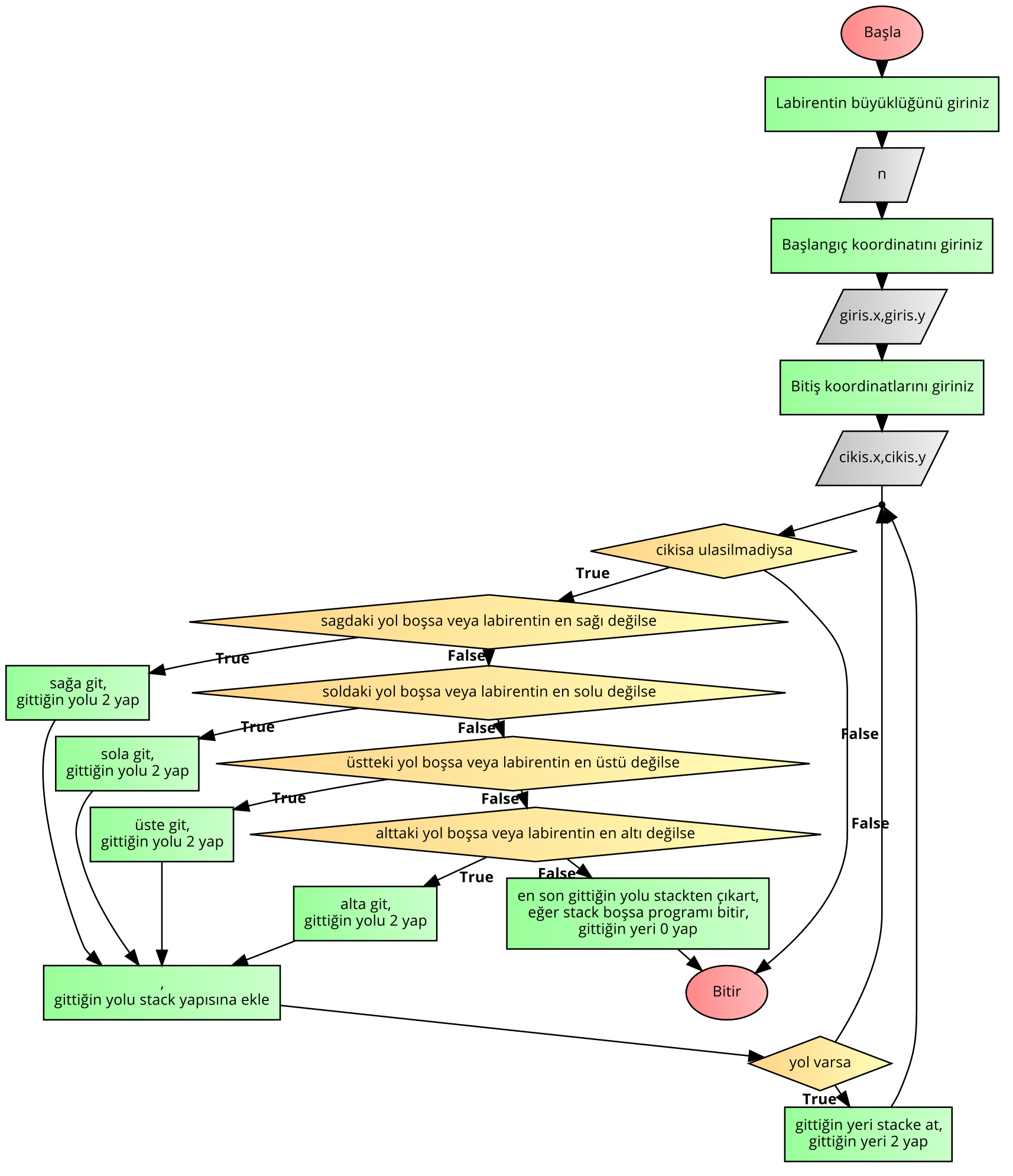


**8.Sonuçlar**

Yapmış olduğumuz programın başarıyla gerçekleştirdiği işlemler:

* Rastgele oluşturulacak olan labirentin boyutu kullanıcı tarafından alınmaktadır. Matrisin boyutu ekranda görünebilecek kadar olmaktadır. Bu değer maksimum 18 olarak belirlenmiştir. 18’den büyük bir değer girilmesi durumunda kullanıcıdan tekrar labirent boyutu girmesi istenmiştir.
* Oluşturulan matrisin elemanları 1 veya 0 olarak rastgele bir şekilde oluşturulmaktadır. Ek olarak; bu matris bağlı liste olarak tutulmaktadır. Rastgele üretilen bu 1 ve 0 değerleri bağlı liste yapısındaki matrise atılmaktadır.
* Üretilen matris konsol ekranı yerine, daha kullanıcı dostu olduğu gerekçesiyle, “graphics.h” kütüphanesi kullanılarak kullanıcıya gösterilmektedir. Bu arayüzde, “graphics.h” kütüphanesinden faydalanılarak renkler ve kutucuklar kullanıldığından dolayı, oluşturulan arayüz daha kullanıcı dostu olmaktadır.
* Programın çalışması sırasında, giriş ve çıkış kapıları kullanıcı tarafından alınmaktadır. Eğer kullanıcının girmiş olduğu kapılar kapalı ise kullanıcının tekrar giriş yapması istenmektedir. Bu sırada, eğer kullanıcı -1 değeri girerse kullanıcıya tekrar labirent gösterilerek giriş ve çıkış koordinatlarını hatırlaması sağlanmaktadır.
* Programın içerisinde, hiçbir yerde dizi yapısı kullanılmamıştır. Programın çalışması sırasında, başta rastgele olarak oluşturulan labirent bağlı liste yapısı içerisine atılarak matris oluşması sağlanmıştır. Programın ileriki kısımlarında ise yığın ve bağlı liste yapıları birlikte kullanılarak, yol bulma sırasında, her gidilen adım tek tek yığın yapısı içerisine atılmıştır. Labirent içerisinde yığın yapısı kullanılarak dolaşılması sağlanmıştır.
* Yığın ve bağlı liste yapısı kullanılarak bulunan yol; en son kullanıcıya gösterilmektedir. Bu yolun kullanıcıya gösterilmesi sırasında; yine “graphics.h” kütüphanesinden yararlanılarak daha kullanıcı dostu bir arayüz içerisinde bulunan yolun kullanıcıya gösterilmesi sağlanmıştır.
* Programdaki tüm grafiksel gösterim gereken yerlerde konsol ekranına göre daha kullanıcı dostu olduğu gerekçesiyle “graphics.h” kütüphanesinden yararlanılmıştır. Bu kütüphane yardımıyla ekstra bir pencere içerisinde konsol ekranında yapılamayacak şekiller ve renkler kullanılmıştır. Burada kullanıcının gözüne hitap edilmesi amaçlanmıştır. Ayrıca konsol ekranındaki, kapıların koordinatlarının kullanıcı tarafından hesaplanması sorunu da grafiksel arayüz sayesinde ekranda program tarafından gösterilmiştir. Kullanıcı bu hesaplama yükünden kurtulmuştur.

Yukarıdaki değerlendirmeler ışığında, yapmış olduğumuz program, proje sırasında bizlerden istenen tüm adımları yerine getirmekte olup, hiçbir eksik fonksiyonu bulunmamaktadır. Program, belirlenen hedefe tam anlamıyla ulaşmaktadır.



**9.Referanslar**

<https://www.tutorialspoint.com/c_standard_library/c_function_clock.htm> (Access Date: 19.11.2017)

<http://www.zentut.com/c-tutorial/c-linked-list/> (Access Date: 19.11.2017)

<http://www.learn-c.org/en/Linked_lists> (Access Date: 19.11.2017)

<https://stackoverflow.com/questions/26583717/how-to-scanf-only-integer> (Access Date: 25.11.2017)

<https://stackoverflow.com/questions/28917175/accepting-only-integer-in-c> (Access Date: 25.11.2017)

<https://stackoverflow.com/questions/14099473/how-to-scanf-only-integer-and-repeat-reading-if-the-user-enter-non-numeric-chara> (Access Date: 25.11.2017)

<http://winbgim.codecutter.org/> (Access Date: 26.11.2017)

<https://www.youtube.com/watch?v=FxCdMM9H66I> (Access Date: 26.11.2017)

<https://www.youtube.com/watch?v=XFounOguGYA> (Access Date: 26.11.2017)

<https://www.youtube.com/watch?v=fWmis4uVCHg> (Access Date: 26.11.2017)

<http://www.codeincodeblock.com/2011/09/how-to-setup-winbgim-library-in.html> (Access Date: 26.11.2017)

<http://www.codewithc.com/how-to-include-graphics-h-in-codeblocks/> (Access Date: 26.11.2017)

<http://www.programmingsimplified.com/c/graphics.h> (Access Date: 26.11.2017)