**Labirent Oyunu**



#define kullanarak labirentimizin satır ve sutün uzunluğunu belirliyoruz. Struct yapısını kullanarak labirentteki pozisyonlarımızı point’e aktarıyoruz.



Char ve dizi kullanarak labirentimizi oluşturuyoruz. Stack kütüphanesi kullanarak konumlarımızı saklayacak point tipinde yığın oluşturuyoruz. Giriş ve çıkış konumlarını belirlemek için point tipinde entrance ve exit değişkenleri oluşturup bu değişkenlere x ve y değerleri atıyoruz.

Güncel konumumuzu takip etmek için currentPoint değişkeni oluşturup bu değişkene entrance değerini atıyoruz

Duvar ve yol değişkenlerimizi belirledikten sonra gidilen yerlere işaret bırakmak için explored adında bir değişken oluşturuyoruz.

Konsolda gösterme ve işlemler için iki tane labirent oluşturuyoruz ve bu labirentlere orijinal labirentin değerlerini atıyoruz.



Çıkış noktası bulunana kadar çalışacak bir döngü açıyoruz ve gidebileceğimiz yönleri belirliyoruz.



Eğer bir üstümüz gidilebilecek bir yolu belirtiyorsa güncel konumumuzu yığına ekleyip, gidilmiş olarak işaretliyoruz ve bir üste gidiyoruz.



Aynı mantığı kullanarak diğer yönleri kontrol ediyoruz. Eğer kontrol ettiğimiz yönlerde duvar var ise labirentteki konumumuzu duvar olarak işaretleyip, güncel konumumuza yığının tepe elemanını atıyoruz ve yığından bir eleman çıkartıyoruz.



Point değerlerini saklayan ve (satın \* sütun) uzunluğunda olan bir dizi oluşturuyoruz. Döngü kullanarak yığındaki değerleri path dizisine aktarıyoruz.

Labirentte dolaştığımızı göstermek için döngü açıyoruz ve tempPoint değişkenimize path dizisindeki değerleri aktarıyoruz.

Daha sonra konsolda göstermek için oluşturduğumuz labirentte olduğumuz konuma \* koyup, labirenti konsolda gösteriyoruz. 300 milisaniye bekledikten sonra konsolu temizleyip önceki konumumuza boşluk koyuyoruz.



Labirentin sonuna ulaştığımızda labirenti gösterip, çıkış yapıldı mesajını veriyoruz.



**Kaynaklar**

<https://stevernote.com/a-simple-maze-solver-in-c/>

<https://www.geeksforgeeks.org/stack-in-cpp-stl/>