SAKARYA UYGULAMALI BİLİMLER ÜNİVERSİTESİ

BİLGİSAYAR MÜHENDİSLİĞİ

VERİ YAPILARI VE ALGORİTMALAR DERSİ PROJE RAPORU

HAZIRLAYANLAR:

23010903055 - FATMA YAŞAR

23010903049 - SÜMEYYE GÜL

Projenin Amacı

Bu projede, MxN boyutlarında bir alanda bulunan bir labirentin başlangıç noktası ile çıkış noktası arasındaki yolları kullanarak gezinti sağlanması ve çıkış noktasının bulunması hedeflenmiştir.

Yığıt (Stack) veri yapısı birçok problemin çözümünde kullanılan önemli bir veri yapısıdır. Bu projemizde de önemli bir yer almaktadır. Labirent oyunumuz da yaptığımız hamleleri yığıt kullanarak defalarca geri alabiliyoruz.

Proje İçeriği

Projede gerekli çalışmalardan biri, programın derlenmesi ve çalışabilir hale getirilmesidir. Bunun için bir hocanin bize vermiş olduğu makefile dosyası oluşturulmuş ve "mingw32-make" komutuyla çalışabilirlik Dev-C++' da test edilmiştir.

Harita yükleme işlemlerini, Labirent.exe çalıştırılarak "harita.txt"dosyasından labirent verilerini açar. Eğer dosya açılamazsa program sonlandırılması için hata mesajı cerr akışına yazdırılır. Bu, programda bir hata oluştuğunu belirtmek için kullanılan bir çıkış akışıdır. Hata olarak kullanmış olduğumuz 'exit(EXIT_FAILURE)' komutu, program hatalı bir durumla karşılaştığı için bu komutla sonlandırılır.

- Labirentin Okunması ve Yapısı:
Labirent, harita dosyasındaki verileri kullanarak içeriğini oluşturur. İşlem süresince adım atma, engel kontrolü gibi fonksiyonlar uygulanır.

- Yol Bulma Algoritması:
Labirentte bir yol aranırken, geçerli bir adım olmadığında bir önceki adıma geri dönülerek diğer olasılıklar denenir. Bu da stack'in özellikleri sayesindedir. Bu sırada her adımda geçilebilir bir yol tercih edilerek ilerlenir.

Kodlama Detayları

1. Konum ve Yön Tanımları

- Konumlar ‘x’ ve ‘y’ koordinatlarıyla, yönler ise coğrafi yönler ise typedef enum { GUNEY, BATI, KUZEY, DOGU } ile tanımlanmıştır. Projeyi Dev-C++ da yapıldığı için okları = {31, 17, 30, 16}; bu şekilde alamadıktan sonra (‘^, >, v, <’) bu şekile çevrildi.

- ‘konum.cpp’ dosyasında 3 farklı yapıcı fonksiyonlar(Parametreli Yapıcı Fonksiyon),(Parametreli Yapıcı Fonksiyon),(Varsayılan Yapıcı Fonksiyon) , yön değiştirme işlevleri ve yönlerin tersine döndürülmesi gibi önemli özellikler açıklanmıştır. Constructor tanımlamalarında, liste başlatıcı (`:`) tercih edilmiştir. Oyunlarda ve benzeri uygulamalarda nesnelerin veya karakterlerin hareketlerini takip etmek için oldukça faydalıdır. Bu sınıfın sağladığı fonksiyonlar, yön değişimlerini ve hareketleri kolayca yönetmeyi mümkün kılar, böylece kullanıcılar veya oyun karakterleri, hareketlerini doğru şekilde belirleyebilirler.

- Kodun okunabilirliği ve esnekliği için ‘switch-case' yapısı ile yön değiştirme işlevleri yazılmıştır. ‘return *this;’ olarak this’ide burada geçersiz bir yön durumunda aynı konuma dönsün diye kullanıldı.

2. Kütüphaneler ve Veri Yapıları

Projede aşağıdaki kütüphaneler kullanılmıştır:
- `fstream`, `iomanip`, `sstream`, `stdexcept`, `iostream`, `vector`, `windows.h`

- Labirent.hpp, Stack.hpp, Konum.hpp başlık dosyalarında tanımlanan fonksiyonların gerçekleştirilmesini Labirent.cpp, Main.cpp, Konum.cpp ile sağlanmıştır.

- Veri yapılarından stack kullanılarak, algoritmanın geriye dönül ve dallanılabilir yapısı gerçekleştirilmiştir.

3. Labirent Fonksiyonları

`labirent.cpp` dosyasında, labirentin duvarları ve geçiş noktaları tanımlanmıştır. Bu dosya, 'labirent.hpp' başlık dosyasında tanımlanan Labirent sınıfının fonksiyonlarının gerçekleştirilmesini sağlar. Labirent haritasını yükleme, kullanıcı hareketlerini işleme ve labirent içindeki mantık işlemlerini kapsar. Kullanıcı, başlangıç noktasından bitiş noktasına kadar bir yol bulmaya çalışacaktır. Program, engelleri tanır ve her adımda haritayı güncelleyerek kullanıcıya anlık durum bilgisi sunar.

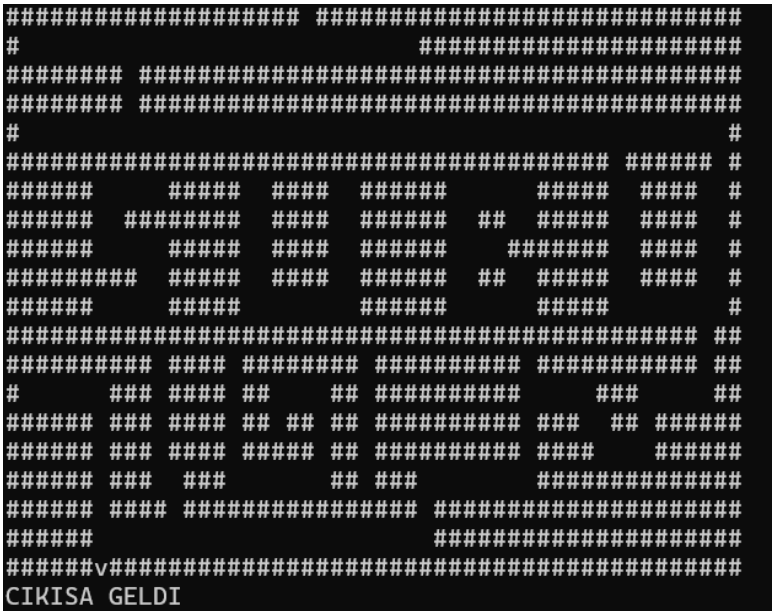
Labirentin genişlik ve yükseklik bilgilerini #define YUKSEKLIK 20 #define GENISLIK 50 Labirent.hpp de gosterildi.

4.Harita Dosyası (Harita.txt)

Harita dosyası her satırda bir karakter dizisi içerir ve her bir karakter şu şekilde anlamlandırılır:

-Engel için='#' Boş alan için='-' kullanıldı.

-Haritamız için hocanın örnek olarak gösterdiği SUBU yazılı harita yapıldı.



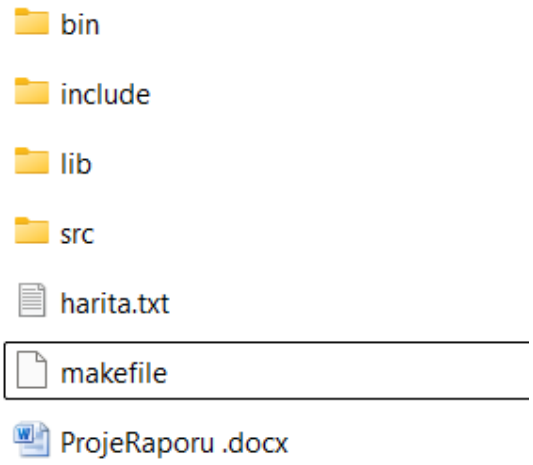
5. Main

Programda, başlangıç konumu ‘(0, 20)’ ve bitiş konumu ‘(19, 6)’ olan bir labirent nesnesi oluşturulmuştur. Bu konumlar, labirentin giriş ve çıkış noktalarını temsil eder. Programda, başlangıçta karakterin hareket edeceği yön, (GUNEY) olarak ayarlanır. Bu, GUNEY yönüyle hareket etmesini sağlar. Program, çıkışa ulaşana kadar bir döngü içinde hareket eder. Her döngüde, program mevcut yönüyle hareket etmeyi dener. Eğer mevcut yönle ilerlemek mümkün değilse, saat yönünde yön değiştirerek yeniden hareket etmeyi dener. Tüm yönler denenip hala ilerleme sağlanamazsa, “labirent->yigit->pop();” komutu ile yığındaki önceki konuma geri dönülür. Geri adım atar böylelikle.

Program, labirent çıkışına ulaşana kadar döngüye devam eder. Çıkışa ulaşılması durumunda, kullanıcıya "CIKISA GELDI" mesajı gösterilir.

Çalışma Performansı

Kodlar, verilen gerekliliklere uygun olarak derlenmiş ve test edilmiştir. Ödevde eksik kalınan bir bölüm bulunmamaktadır. Program, çalışabilir bir durumda olup “makefile” dosyası ve gerekli diğer dosyalar ile birlikte sisteme zıplenerek klasör şeklinde yüklenmiştir.



Sonuç

Bu rapor, labirent çözme oyununu işleyişini açıklamaktadır. Program, başlangıç konumundan bitiş noktasına ulaşmak için yön denemeleri yaparak ve gerektiğinde geri adım atarak labirenti çözmeye çalışır. Kullanıcı, çıkışa ulaştığında program sona erer. Labirent algoritması, başlangıç ve bitiş noktaları arasında başarıyla çalışmış ve ödev gerekliliklerini karşılamıştır. Kodun düzenli ve sistematik bir yapıda yazılmasına özen gösterilmiştir.

Kaynakça

- https://youtu.be/385bQhMPeJ0?feature=shared

- https://github.com/mfadak/DataStructures/tree/main/Week7/Labirent