T.C.

SAKARYA ÜNİVERSİTESİ

BİLGİSAYAR VE BİLİŞİM BİLİMLERİ FAKÜLTESİ

BİLGİSAYAR MÜHENDİSLİĞİ BÖLÜMÜ

PROGRAMLAMA DİLLERİNİN PRENSİPLERİ ÖDEV RAPORU

1.Ödev

G181210058 Burak BALA

SAKARYA

Nisan, 2021

Programlama Dillerinin Prensipleri Dersi

Java ile C++ dosyasını analiz etme

Burak BALA, G181210058 2A

Özet

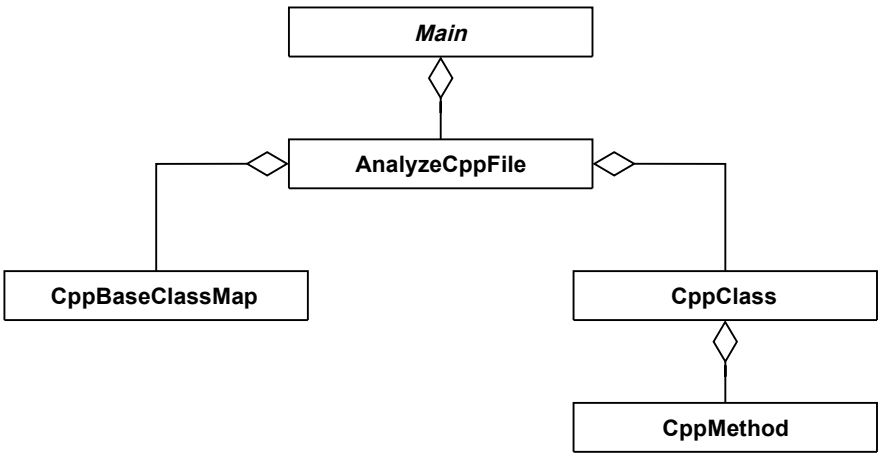
Ödevde ilk önce dosyayı okuyup tek bir string değişkenine atadım. Bu değişkeni kullanarak oluşturduğum regex birçok pattern ile C++ dosyasını analiz edip, uygun sınıflar oluşturup bu sonuçları ilgili nesneleri oluşturmada kullandım ve ilgili nesnelerin print metotlarını kullanarak sonucu yazdırdım. Regex ifadelerini oluşturmak gerçekten zordu, çok fazla ihtimal vardı bu yüzden parçalayarak ve birden çok regex ifadesi kullanarak sonuca ulaştım. Aslında araştırma yaptığımda bu işlemlerin regex ile değil de grammer parser ile yapılmasının daha uygun olduğunu gördüm ama bu konuyu öğrenmek regex öğrenmekten daha uzun ve compiler design konusu sanırım. Main içinde AnalyzeCppFile sınıfından bir nesne oluşturarak bütün işlemeler yapılıyor. AnalyzeCppFile sınıfı içinde CppBaseClassMap, CppClass ve CppMethod sınıflarından uygun regex ifadesini sağlayan yerlerde nesne ürettim. CppBaseClassMap içinde Map veri yapısı ile bir sınıftan kaç kere kalıtım alındığını ve kalıtım alınan sınıfın ismini tuttum böylelikle uygun isim görüldüğünde isme ait verinin tuttuğu sayı arttı. Tüm regex sorguları tamamlanıp nesne ürettikten sonra nesnelerin içeriklerini yazdırdım. Bu sınıflar arasındaki ilişki Şekil 1'de mevcut.

Bu rapor benim özgün çalışmamdır. Faydalanmış olduğum kaynakları içeresinde belirttim. Her hangi bir kopya işleminde sorumluluk bana aittir.

Anahtar Kelimeler: regex, nesneye dayalı programlama, grammer

1. GELİŞTİRİLEN YAZILIM

İlk önce dosyayı BufferReader ile okudum ve tek bir değişkene atadım daha sonra ilk önce birden fazla olan boşlukları bir boşluğa çevirdim, line comments ve block comments yapılarını sildim, const keyword içeren kısımları sildim, using enum, enum class, enum ve struct kısımları sildim, line feeds kısımları sildim, döngü, if-else ve while kısımları sildim ve en son main bloğu sildim ve tekrardan fazla boşlukları sildikten sonra okunan dosyayı AnalyzeCppFile sınıfında analiz edilebilecek hale getirdim.



Şekil 1. Programın UML Gösterimi

Operator overloading yapıldığında fonksiyon isminde olabilecek karakterler kümesi için operators referansınabaktım C++ için. Kullandığım regex ifadelerinden bazıları:

https://regex101.com/r/5vlvTo/1

https://regex101.com/r/TJFdO8/1

https://regex101.com/r/VsyiKT/1

https://regex101.com/r/OgLyxw/1

2. ÇIKTILAR

İlk önce ödev dosyasındaki örnek C++ dosyasını analiz ettim, daha sonra karışık bir dosyada denedim ve gönderdiğim dosyada da karışık olan var hazırda. Tek bir noktada yanlış var, örneğin Tekne(int* a[], int[] b[]) olduğunda parametreleri int*, int olarak alıyor. Bu kısmı tek parametre olunca düzelttim birçok parametre için de uygulayacaktım ama ödeve çok zaman ayırdım için vizelere çalışmak için düzeltmedim.

```
Program.cpp dosyasının analiz sonucunun bir kısmı

Class: Kisi
        Kisi
                Parameters: 3 (string, int, double)
                Return type: Object Address
        Kisi
                Parameter: 0
                Return type: Object Address
        getIsim
                Parameter: 0
                Return type: string
        getDogumYili
                Parameter: 0
                Return type: int
        getKilo
                Parameter: 0
                Return type: double
        YemekYe
                Parameter: 1 (double)
                Return type: void
        Kos
                Parameter: 1 (double)
                Return type: void
```

3. SONUÇ

Programlama dillerinin grammer yapısının etkisini regex ifadelerini kullanırken daha iyi anlamış oldum.

Referanslar

[1] cppreference.com

[2] regex101.com

[2] geeksforgeeks.org

[3] visual-paradigm.com

[4] stackoverflow.com

[5] Regular Expressions (Regex) Tutorial

[6] /Reg(exp){2}lained/: Demystifying Regular Expressions