PROJE RAPORU

Derleyici Tasarım

Grup Üyeleri:

Zeki Ahmet BAYAR - 030117092

Ahmet BİÇER - 030718040

Tuğba FIÇICI - 030116005

Süleyman ÖZER – 030117078

# Projenin Amacı:

Derleyici tasarımı dersinde öğrenmiş olduğumuz konseptleri kullanarak sıfırdan bir yorumlayıcı tasarlanması ve kodlanmasıdır. Aşağıda verilen gramere göre yorumlayıcı tasarlanmıştır.

P → {C} '.'

C → I | W | A | Ç | G

I → '[' E '?' C{ C } ':' C{ C } ']' | '[' E '?' C{C} ']'

W → '{' E '?' C{C} '}'

A → K '=' E ';'

Ç → '<' E ';'

G → '>' K ';'

E → T {('+' | '-') T}

T → U {('\*' | '/' | '%') U}

U → F '^' U | F

F → '(' E ')' | K | R

K → 'a' | 'b' | … | 'z'

R → '0' | '1' | … | '9'

# Proje Hakkında:

Kullanılan yazılım dili Python programlama dilidir. Yapılan işler aşağıda sıralandığı gibidir.

1. Giriş metninin lexical analizinin yapılması,

Verilen giriş metninin stack’e atılır. Tokenlerin tek tek kullanılması ve sıradaki tokenin çekilmesi için stack yapısı kullanılıştır. getToken() fonksiyonu yazılarak stack en üstünde bulunan token çıkartılıp işlenmek için return edilir.

1. Verilen gramere göre terminal olmayan semboller için fonksiyonlar oluşturulmuştur. Bu fonksiyonlar gramerde belirtildiği üzere recursive(yinelemeli) olarak tanımlanmıştır.
2. İşlemlerin göz önünde olması için ayrı bir stack oluşturulmuş ve yapılan işlemler bu stack üzerinden takip edilmiştir.
3. Gramere uygun görülen giriş metninin hangi fonksiyonlardan geçtiğini göstermek amacı ile geçme sırasını tutan bir stack ayrıca yer verilmiştir. Her bir fonksiyona girildiğinde fonksiyon adı sıralama stack’ine eklenir.
4. İşlemlerin sonunda ayrıştırma ağacı konsol üzerinden çizdirilmiştir.
5. Sembollere göre hata kontrolleri eklenmiştir.

# Fonksiyonlar:

Her bir fonksiyonun ne iş yaptığı aşağıda açıklanmıştır.

### P():

P fonksiyonu global tokenin “.” olup olmadığını kontrol eder. Eğer noktaysa programın doğru çalıştığı anlamına gelir ve programı bitirir. Eğer nokta değilse C() fonksiyonunu çağırır.

### C():

C fonksiyonu atama, girdi alma, çıktı verme, while ve if işlemlerini gelen tokene göre çağırır. Eğer token “$” ise ve stack’lerde “$” sembolü var ise tokene “.” işaretini atar.

### I():

I fonksiyonu if işlemlerinin gerçekleştirmek için kullanılır. İlk olarak E() fonksiyonu çağırılır. Yeni token alınır ve “?” sembolü olup olmadığı kontrol edilir. Kontrol sonucunda if-then işlemi için C() fonksiyonu çağırılır. İf-else işlemi için C() fonksiyonu çağırıldıktan sonra “:” kontrolü sağlanır ve yeniden C() fonksiyonu çağırılır. Çağırılan C fonksiyonları birden fazla kez çağırılabilir.

### W():

W fonksiyonu while işlemlerini gerçekleştirmek için kullanılır. Gramere göre ilk olarak E() fonksiyonu çağırılır. Yeni token alınarak “?” sembolü kontrol edilir ve buna göre C() fonksiyonu çağırılır. Buradaki C fonksiyonu birden fazla çağırılabilir.

### A():

A fonksiyonu atama işlemleri için kullanılır. İlk olarak gelen tokenin harf olduğu varsayıldığı için K() fonksiyonu çağırılır. Sonraki token alınarak “=” sembolü kontrol edilir ve E() fonksiyonu çağırılır.

### noktaliC():

Ç fonksiyonu çıktı işlemleri için kullanılır. Yeni token alındıktan sonra E() fonksiyonu çağırılır.

### G():

G fonksiyonu girdi işlemleri için kullanılır. Yeni token alındıktan sonra K() fonksiyonu çağırılır.

### E():

E fonksiyonunda aritmetik ifade işlemleri yapılır. İlk token alınıp T() fonksiyonu çağırıldıktan sonra gelen tokenin “+” ve “-“ olması kontrol edilir. Eğer gelen token bu sembollerden biriyse E() fonksiyonu yeniden çağırılarak işlemlerin tekrarlanması sağlanır. Bu şekilde fonksiyon yinelemeli olarak tanımlanır.

### T():

T fonksiyonu çarpma-bölme-mod işlemleri için kullanılır. E fonksiyonunda olduğu gibi T yerine U fonksiyonu çağırılır ve bir sonraki tokenin “\*”, “/” ve “%” olup olmadığı kontrol edilir. T fonksiyonuda yinelemeli olarak tanımlanmıştır.

### U():

U fonksiyonu üslü ifadeler için oluşturulmuştur. F() fonksiyonu çağırılır. Gelen tokenin “^” sembolü olması kontrol edilir. Eğer “^” sembolüyse üslü ifade işlemi için U() fonksiyonu çağırılır. Değilse işlem üslü ifade olmaksızın tamamlanır.

### F():

F fonksiyonu gruplama işlemlerinde kullanılır. Buradan 3 farklı fonksiyon çağırılabilir. Eğer gelen token küçük harf ise K() fonksiyonu, rakam ise R() fonksiyonu, eğer “(“ ise E() fonksiyonu çağırılır.

### K():

Gelen tokenin küçük harf olması kontrol edilir ve işlemler stack’inde harf ile “K” yer değiştirilir.

### R():

Gelen tokenin rakam olması kontrol edilir ve işlemler stack’inde harf ile “R” yer değiştirilir.

# İşlevler:

### Ağaç Oluşturma:

Programdaki fonksiyonların çağırılma sırası ağacın dallarını tam anlamıyla temsil ettiği için fonksiyonu çağıran fonksiyonun ismi python “sys kütüphanesi” kullanılarak çekilmiştir. Başka bir python kütüphanesi olan “treelib kütüphanesi” ağacın görsel kısmını oluşturmak için kullanılmıştır. Ağaçtaki her bir yaprak aşağıdaki fonksiyon kullanılarak ağaca eklenmiştir.

* tree.create\_node(functionName, childID, parent=parentID)

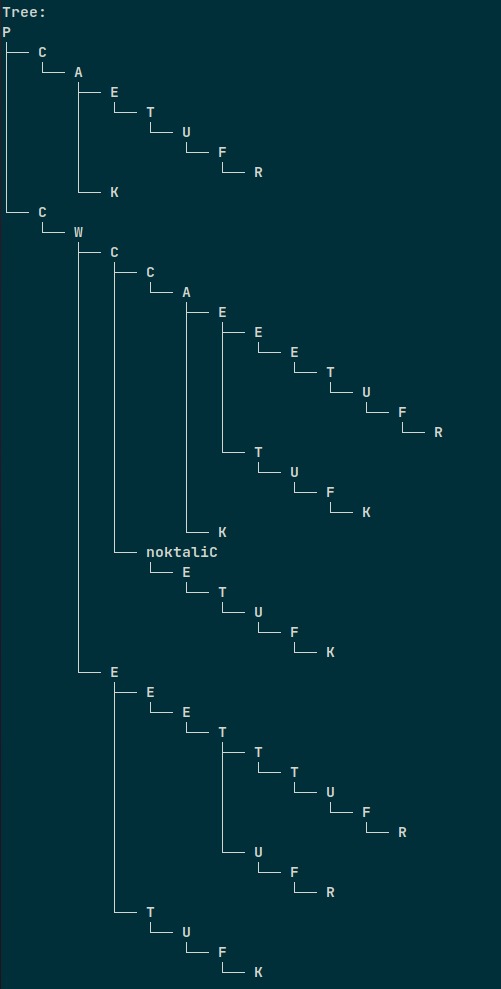
Aynı fonksiyon birden fazla kez çağırılabileceği için her bir fonksiyonun kaçıncı seviyede olduğunu göstermek amacı ile fonksiyona ait bir sayaç kullanılmaktadır. Eğer yinelemeli fonksiyon ise parentID’nin doğru gelmesi için sayacın bir eksiği kullanılmıştır.

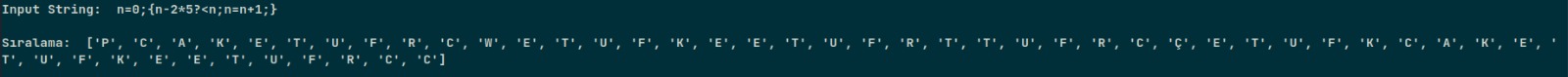
### Stack Yapısı:

İşlemlerin stack üzerinde gösterilmesi için oluşturulmuştur. Burada amaç tokenin stack’e atıldıktan sonra gramere uygun şekilde “reduce” işleminin sağlanmasıdır. “Reduce” işlemine örnek olarak; Stack içeriği [‘$’,’’K,’=’,’E’] şeklindeyken atama fonksiyonu sonrasında son üç sembolün çıkartılarak yerine A eklenmesidir. A->K=E işlemi yapılmış olur. Her bir fonksiyon için gramere uygun olarak bu işlemler yapılmıştır.

Program Çıktısı:

Ödev dosyasında verilen örneğe göre oluşturulan çıktı:







Herhangi bir hata için oluşan çıktı:

