**Yorumlayıcı Tasarım Projesi**

**Proje Amacı ve kapsamı**

Bu projede amaç aşağıda verilen context-free grameri kullanarak bir derleyici yazmaktır. Verilen gramer rekursiv yukarıdan-aşağıya doğru çalışan bir parserin ayrıştırabileceği şekle sokulmuştur.

Aşağıda projede neler yapılması gerektiği ve açıklamalar verilmiştir.

* Gramer genel olarak Backus-Naur formunda verilmiştir.
* Gramerdeki büyük harfler terminal-olmayan sembolleri ve tek tırnak içinde verilenler terminal sembolleri gösterir. Terminal-olmayan semboller gramerin tanımında kullanılır, gerçek programda (aşağıda örneği verilimiştir) kullanılmazlar. Tek tırnak içindeki her sembol gerçek programda aynen görülebilirler/kullanılabilir.
* Gramerde kullanılan { ve } sebmolleri hem terminal sembolü olarak kullanılmış (tek tırnak içindeyse) hem de terminal olmayan sembolün sıfır veya daha fazla tekrar edeceği anlamında kullanılmıştır (tek tırnaksız). Aynı şekilde ( ve ) sembolleri de hem gruplama amacıyla (tek tırnaksız) hem de terminal sembolü olarak kullanılmıştır (tek tırnak içindeyse).
* Gramerin sol tarafında terminal olmayan semboller, sağ tarafında bu terminal olmayan sembollerin nelere gidebileceği verilmiştir. Sağ tarafta terminal olmayan ve terminal sembollerin karışımı görülebilir. Sol ve sağ taraf aradaki okla beraber bir gramer kuralını oluşturur.
* Her kuralın sol ve sağ tarafı birbirinden → sembolüyle ayrılmıştır. Ayrıca satır sayısını azaltmak için birden fazla kural aynı satırda verilmiştir. Örneğin C (Cümle) terminal-olmayan sembolü sağında beş farklı terminal-olmayan sembole gitmiştir. Dolayısıyla her kuralın sağındaki dikey çubuk (|) “veya” anlamı taşır. Yani bir program cümlesi aslında bir şartlı cümle (I) veya döngü (W) vs olabilir.
* P terminal-olmayan kuralının sonundaki terminal sembol (tek tırnak içindeki nokta) programın bitişini temsil ediyor.
* I (IF) terminal-olmayan kuralının içindeki ? ve : terminal sembolleri bazı programlama dillerinde kullanılan kısa şartlı cümlecik yazım yönteminden alıntıdır. Dolayısıyla şartlı cümlelerin yazımını kolaylaştırarak programı kısaltır. I terminal olmayan kuralın ilk kısmı if-then-else yapısını ikincisi ise if-then yapısını temsil eder.
* I (IF) ve W (While döngüsü) terminal-olmayan kuralının Boolean kısımları (kuraldaki E’ler) C dilinde olduğu gibidir. 0 (sıfır) değeri “yanlış”ı başka herhangi bir değer “doğru”yu gösterir.

Aşağıda yorumlayıcısını yazacağınız dilin grameri ve bu gramerdeki herbir terminal ve terminal-olmayan sembolün anlamlarını verilmiştir.

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'

P: program

C: Cümle

I: IF cümlesi

W: While döngüsü

A: Atama cümlesi

Ç: Çıktı cümlesi

G: Girdi cümlesi

E: Aritmetik İfade

T: Çarpma-bölme-mod terimi

U: Üslü ifade

F: Gruplama ifadesi

K: Küçük harfler

R: Rakamlar

**Rekursiv yukarıdan-aşağıya doğru çalışan parserin ve yorumlayıcının yazılımı**

1. İstediğiniz programlama dilini kullanabilirsiniz.
2. ‘Regular expression’ları kullanamazsınız.
3. Başlangıçta lexical analiz yapmanız gerekmektedir. Yani program metnini tokenlerine ayırmanız ilk adımınız olacaktır. Bu işlemi ve parser yazımına ve yorumlayıcıya odaklanmanızı kolaylaştırmak için program degişkenleri tek küçük harf olarak kullanılmış ve bütün sayılar tek rakamdan oluşmuştur. Bir getToken() fonksiyonu yazarak program metninden bir sonraki tokeni isteyebilirsiniz.
4. Recursiv yukarıdan-aşağıya doğru parser yazarken gramerdeki herbir terminal-olmayan sembol için bir fonksiyon yazacaksınız. Dilerseniz terminal olmayan sembolle aynı ismi taşıyan bir fonksiyon yazarak olası bir karışıklığın önüne geçebilirsiniz. Yorumlayıcınız ilk token’i okuyup P fonksiyonunuzu çağırır (Herbir terminal olmayan sembol için aynı ismi taşıyan bir fonksiyon yazdığınızı kabul edin). P fonksiyonu P kuralının program metni tarafından takip edilip edilmediğini kontrol eder ve aynı zamanda oluşan her cümleyi çalıştırır. Örneğin bir atama cümlesinin sonuna gelindiyse yorumlayıcı içinde tuttuğu program değişkenine değeri atar. C halledildikten sonra programın son sembolü nokta (.) olmalıdır. Peki noktadan önce neler görülebilir? C sembolü I, W, A, Ç veya G sembollerinden birine gidebilir. Bunların hepsi terminal olmayan sembollerdir ve nihayetinde terminal olan sembol ve sembollere gitmek zorundadır çünkü programda token olarak okunan her şey gramerde terminal olan bir semboldür. Dolayısıyla I '[' terminal sembolüne, W '{' sembolüne, A K’ye ve dolayısıyla herhangi bir (terminal harfe), Ç '<' sembolüne ve G '>' sembolüne gidecektir. Kısacası C fonksiyonu içinde okunan token’in '[', '{' , herhangi bir (terminal) harfe, '<' veya '>' sembolünden biri olmak zorundadır. C opsiyonel olduğu için nokta terminal sembolü de (.) C’de legal bir semboldür (boş bir program).
5. Yorumlayıcınız herhangi bir ayrıştırıcı ağacını açık bir şekilde oluşturmayacaktır. Bu ağaç fonksiyonlar içinde aslında gömülü bir şekilde mevcuttur. Programınız ekrana sadece çıktı sembolü kullanıldığında yazacaktır. Dolayısıyla program çıktısını görmek için '<' sembolü ve sonrasında aritmetik ifade programda yer almalıdır.
6. Aşağıdaki örnek yukarıdaki gramerden türetilebilecek bir programdır. Yorumlayıcınızı yazmaya başlamadan önce aşağıdaki programın yukarıda verilen gramerden türetilebileceğinden emin olun. Bunun için P’den başlayarak uygun gramer kurallarının sağ taraflarını kullanarak bütün terminal olmayan sembolleri uygun olan terminal sembollerle değiştirerek aşağıdaki programı elde edin. Sonrasında da yorumlayıcınızı yazmaya başlayabilirsiniz.

n = 0;

{ n - 2\*5 ?

< n;

n = n + 1;

}

.

Lütfen projeyi hazırlarken aşağıdaki şartları göz önünde bulundurun,

1- C#, C++,Java veya C kullanılmalıdır. Başka bir dil kullanmak isterseniz onayım için e-postamı kullanabilirsiniz.

2- Elde ettiğiniz sonuçları ve tartışmayı içeren bir rapor yazmanız gerekir.

3- Proje sunum gününde bilgisayarlarınızı ve raporunun basılı kopyasını yanınızda getiriniz. Ayrıca "rapor & kod dosyasını" CD’ye kaydedip tüm dosyaları sisteme yükleyiniz.

4- Dosya adı: Dönemprojesi\_Ad\_Soyad\_Öğrenci.

Sunum haftaları:

12, 13 ve 14. haftalar.

Saygılarımla,

Dr. Öğr. Üyesi Ali HAMİTOĞLU